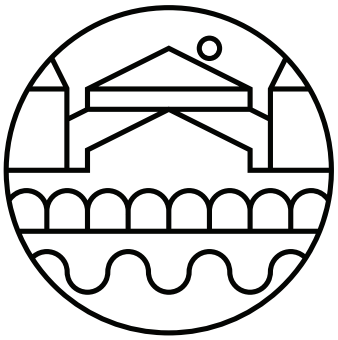
XXXV
SIMPÓSIO BRASILEIRO DE
REDES DE COMPUTADORES
E SISTEMAS DISTRIBUÍDOS

15 a 19 de maio de 2017
Belém - Pará/Hotel Princesa Louçã

ANAIS SBRC 2017



XXXV
SIMPÓSIO BRASILEIRO DE
REDES DE COMPUTADORES
E SISTEMAS DISTRIBUÍDOS
15 a 19 de maio de 2017
Belém - Pará/Hotel Princesa Louçã

Anais do SBRC 2017
Trilha Principal e Salão de Ferramentas

Editora
Sociedade Brasileira de Computação (SBC)

Organização
Michele Nogueira Lima (UFPR)
Edmundo Roberto Mauro Madeira (UNICAMP)
Fabio Luciano Verdi (UFSCar)
Antônio Jorge Gomes Abelém (UFPA)
Eduardo Coelho Cerqueira (UFPA)

Realização
Sociedade Brasileira de Computação (SBC)
Universidade Federal do Pará (UFPA)
Laboratório Nacional de Redes de Computadores (LARC)

Capa: Catarina Nefertari (PCT-UFPA)
Produção Editorial: Denis Lima do Rosário (UFPA)

Cópias Adicionais:
Sociedade Brasileira de Computação (SBC)
Av. Bento Gonçalves, 9500- Setor 4 - Prédio 43.412 - Sala 219
Bairro Agronomia - CEP 91.509-900 - Porto Alegre - RS
Fone: (51) 3308-6835
E-mail: sbc@sbc.org.br

XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (35: 2017: Belém, Pa).

Anais / XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos; organizado por Antônio Jorge Gomes Abelém, Eduardo Coelho Cerqueira, Edmundo Roberto Mauro Madeira, Michele Nogueira Lima, Fabio Luciano Verdi - Porto Alegre: SBC, 2017

1116 p. il. 21 cm.
Vários autores
Inclui bibliografias
ISSN: 2177-496X

1. Redes de Computadores. 2. Sistemas Distribuídos. I. Abelém, Antônio Jorge Gomes II. Cerqueira, Eduardo Coelho III. Lima, Michele Nogueira IV. Madeira, Edmundo Roberto Mauro V. Verdi, Fabio Luciano VI. Título.

Sociedade Brasileira da Computação

Contato
Av. Bento Gonçalves, 9500
Setor 4 - Prédio 43.412 - Sala 219
Bairro Agronomia
91.509-900 – Porto Alegre RS
CNPJ: 29.532.264/0001-78
http://www.sbrc.org.br

Laboratório Nacional de Redes de Computadores (LARC)

Diretora do Conselho Técnico-Científico
Rossana Maria de C. Andrade (UFC)

Vice-Diretor do Conselho Técnico-Científico
Ronaldo Alves Ferreira (UFMS)

Diretor Executivo
Paulo André da Silva Gonçalves (UFPE)

Vice-Diretor Executivo
Elias P. Duarte Jr. (UFPR)

Membros Institucionais
SESU/MEC, INPE/MCT, UFRGS, UFMG, UFPE, UFCG (ex-UFPB Campus Campina Grande), UFRJ, USP, PUC-Rio, UNICAMP, LNCC, IME, UFSC, UTFPR, UFC, UFF, UFSCar, IFCE (CEFET-CE), UFRN, UFES, UFBA, UNIFACS, UECE, UFPR, UFPA, UFAM, UFABC, PUCPR, UFMS, UnB, PUC-RS, PUCMG, UNIRIO, UFS e UFU.

Contato
Universidade Federal de Pernambuco - UFPE
Centro de Informática - CIn
Av. Jornalista Anibal Fernandes, s/n
Cidade Universitária
50.740-560 - Recife - PE
http://www.larc.org.br

Organização do SBRC 2017

Coordenadores Gerais
Antônio Jorge Gomes Abelém (UFPA)
Eduardo Coelho Cerqueira (UFPA)

Coordenadores do Comitê de Programa
Edmundo Roberto Mauro Madeira (UNICAMP)
Michele Nogueira Lima (UFPR)

Coordenador de Palestras e Tutoriais
Edmundo Souza e Silva (UFRJ)

Coordenador de Painéis e Debates
Luciano Paschoal Gaspary (UFRGS)

Coordenadores de Minicursos
Heitor Soares Ramos (UFAL)
Stênio Flávio de Lacerda Fernandes (UFPE)

Coordenadora de Workshops
Ronaldo Alves Ferreira (UFMS)

Coordenador do Salão de Ferramentas
Fabio Luciano Verdi (UFSCar)

Comitê de Organização Local
Adailton Lima (UFPA)
Alessandra Natasha (CESUPA)
Davis Oliveria (SERPRO)
Denis Rosário (UFPA)
Elisangela Aguiar (SERPRO)
João Santana (UFRA)
Josivaldo Araújo (UFPA)
Marcos Seruffo (UFPA)
Paulo Henrique Bezerra (IFPA)
Rômulo Pinheiro (UNAMA)
Ronedo Ferreira (META)
Thiêgo Nunes (IFPA)
Vagner Nascimento (UNAMA)

Comite Consultivo
Allan Edgard Silva Freitas (IFBA)
Antonio Alfredo Ferreira Loureiro (UFMG)
Christian Esteve Rothenberg (UNICAMP)
Fabíola Gonçalves Pereira Greve (UFBA)
Frank Augusto Siqueira (UFSC)
Jussara Marques de Almeida (UFMG)
Magnos Martinello (UFES)
Antonio Marinho Pilla Barcellos (UFRGS)
Moisés Renato Nunes Ribeiro (UFES)
Rossana Maria de Castro Andrade (UFC)

Mensagem dos Coordenadores Gerais

Sejam bem-vindos ao 35o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2017) e a acolhedora cidade das mangueiras - Belém / Pará.

Organizar uma edição do SBRC pela segunda vez no Norte do Brasil é um desafio e um privilégio por poder contribuir com a comunidade de Redes de Computadores e Sistemas Distribuídos do Brasil e do exterior. O SBRC se destaca como um importante celeiro para a discussão, troca de conhecimento e apresentação de trabalhos científicos de qualidade.

A programação do SBRC 2017 está diversificada e discute temas relevantes no cenário nacional e internacional. A contribuição da comunidade científica brasileira foi de fundamental importância para manter a qualidade técnica dos trabalhos e fortalecer a ciência, tecnologia e inovação no Brasil.

Após um cuidadoso processo de avaliação, foram selecionados 77 artigos completos organizados em 26 sessões técnicas e 10 ferramentas para apresentação durante o Salão de Ferramentas. Além disso, o evento contou com 3 palestras e 3 tutoriais proferidos por pesquisadores internacionalmente renomados, 3 painéis de discussões e debates, todos sobre temas super atuais, 6 minicursos envolvendo Big Data, sistemas de transportes inteligentes, rádios definidos por software, fiscalização e neutralidade da rede, mecanismos de autenticação e autorização para nuvens computacionais e comunicação por luz visível, bem como 10 workshops.

O prêmio “Destaque da SBRC” e uma série de homenagens foram prestadas para personalidades que contribuíram e contribuem com a área. O apoio incondicional da SBC, do LARC, do Comitê Consultivo da SBRC e da Comissão Especial de Redes de Computadores e Sistemas Distribuídos da SBC foram determinantes para o sucesso do evento. A realização do evento também contou com o importante apoio do Comitê Gestor da Internet no Brasil (CGI.br), do CNPq, da CAPES, do Parque de Ciência e Tecnologia Guamá, da Connecta Networking, da Dantec Telecom, da RNP e do Google. Nosso especial agradecimento à Universidade Federal do Pará (UFPA) e ao Instituto Federal do Pará (IFPA) pelo indispensável suporte à realização do evento.

Nosso agradecimento também para os competentes e incansáveis coordenadores do comitê do programa (Michele Nogueira/UFRP – Edmundo Madeira/UNICAMP), aos coordenadores dos minicursos (Stênio Fernandes/UFPE – Heitor Ramos/UFAL), ao coordenador dos workshops (Ronaldo Ferreira/UFMS), ao coordenador de painéis e debates (Luciano Gaspary/UFRGS), ao coordenador do Salão de Ferramentas (Fabio Verdi/UFSCar) e ao coordenador de palestras e tutoriais (Edmundo Souza e Silva/UFRJ). Destacamos o excelente trabalho do comitê de organização local coordenado por Denis do Rosário.

Por fim, desejamos a todos uma produtiva semana em Belém.

Antônio Abelém e Eduardo Cerqueira
Coordenadores Gerais do SBRC 2017

Mensagem dos Coordenadores do Comitê de Programa do SBRC 2017

O SBRC é um evento científico nacional consolidado e de reconhecida qualidade. Seu histórico de submissões remete a força da comunidade que realiza pesquisa em redes de computadores e sistemas distribuídos. Na sua 35a. edição, foram submetidos 247 trabalhos completos, dos quais 79 foram aceitos para apresentação e publicação, correspondente a 32% de taxa de aceitação. Todos os 247 trabalhos passaram por uma avaliação criteriosa feita pelos 121 membros do Comitê de Programa e por 129 revisores associados a eles. Cada artigo foi avaliado por, pelo menos, três especialistas na área, sendo que a grande maioria dos trabalhos (mais de 95%) teve quatro pareceres.

Como tem ocorrido em várias edições do SBRC, após a etapa inicial de avaliação dos trabalhos, os autores tiveram a oportunidade de submeter um rebuttal (réplica), fazendo os devidos esclarecimentos técnicos. Em seguida, cada revisor foi incentivado a avaliar o rebuttal e discutir as avaliações com os demais revisores. Nesse processo, contamos com a ajuda de revisores líderes responsáveis por convergir a discussão e chegar a uma recomendação final. Especial atenção foi dada aos artigos na "zona cinza", ou seja, onde havia dúvidas sobre a aceitação ou a rejeição. Ao discutir o ponto de corte, os membros do comitê foram consultados sobre a taxa de aceitação. Por uma questão de transparência, os trabalhos submetidos pelos coordenadores do comitê de programa foram avaliados de forma totalmente independente (externo ao JEMS), e agregados apenas após a definição do ponto de corte. Finalizado todo o processo de seleção, os resultados foram consolidados, discutidos e homologados pelos membros do Comitê de Programa. Posteriormente, foi organizada uma comissão formada por alguns membros do Comitê de Programa para escolher os trabalhos destaque desta edição do evento.

O resultado final é uma programação técnica que remete muito bem a diversidade e o vigor das iniciativas de pesquisa em curso nas universidades, centros de pesquisa e empresas do País. Os trabalhos aceitos foram divididos em 26 sessões técnicas, onde serão discutidos modelos, abordagens, tecnologias, aplicações e serviços, recentes e emergentes, das diversas áreas de redes de computadores e sistemas distribuídos. As sessões técnicas compreendem desde Redes Sem Fio, Redes Móveis e Dinâmicas, Redes Celulares, Redes de Sensores e Redes Corporais, Redes Tolerantes a Atrasos, Redes Ópticas, Redes Definidas por Software, Virtualização de Funções de Rede e Redes Veiculares, até Computação nas Nuvens, Sistema P2P e Tolerância a Falhas, passando pela Internet das Coisas, Redes Sociais e Aplicações de E-commerce, e Redes Centradas em Conteúdo.

Gostaríamos de expressar os nossos sinceros agradecimentos aos membros do Comitê de Programa e revisores pelo árduo trabalho feito e pela pronta resposta às nossas várias chamadas e solicitações. A grande maioria dos pareceres foi de excelente qualidade, demonstrando grande objetividade, imparcialidade e com boas sugestões para os autores. A publicação de um trabalho científico só se realiza com a ativa participação de revisores sérios e comprometidos com esse processo. Somente com esse trabalho cuidadoso de avaliação e seleção de artigos que o alto nível de qualidade do SBRC pode ser mantido.

Queremos fazer um agradecimento especial aos coordenadores gerais desta 35a. edição do SBRC, professores Antônio Jorge Gomes Abelém e Eduardo Coelho Cerqueira (ambos UFPA), pela confiança depositada em nosso trabalho e pelo pronto apoio durante todo o processo.

Agradecemos também aos autores que submeteram trabalhos ao SBRC 2017, fundamentais para seu sucesso.

Desejamos a todos os participantes do SBRC 2017 uma semana bastante produtiva e rica em discussões técnicas instigantes bem como conversas inspiradoras para novos projetos e cooperações. Não deixem também de conhecer um pouco da história e cultura de Belém do Pará!

Michele Nogueira Lima e Edmundo Roberto Mauro Madeira

Coordenadores do Comitê de Programa

Mensagem do Coordenador do Salão de Ferramentas 2017

Sejam todos bem-vindos ao Salão de Ferramentas do SBRC 2017

Fazer pesquisa em Redes de Computadores e Sistemas Distribuídos em nosso dia-a-dia é algo tremendo! Entender protocolos, implementar soluções, definir arquiteturas e realizar medições são alguns exemplos de atividades que desenvolvemos em nossa área. Os resultados de muitas dessas atividades não somente geram artigos científicos mas também ferramentas, muitas delas muito relevantes para a comunidade científica. Neste sentido, o Salão de Ferramentas do SBRC é uma oportunidade para que pesquisadores apresentem suas soluções que muito os auxiliaram ao longo do ano em suas pesquisas. As ferramentas nos permitem reproduzir cenários de maneira escalável e com mais rapidez, gerenciar situações automaticamente, configurar sistemas e apresentar resultados, dentre diversas outras possibilidades.

No ano de 2017, tivemos 27 ferramentas registradas e 18 submetidas. Dessas, 12 foram aceitas. Considerando o histórico de ferramentas aceitas, este ano posso afirmar que o número foi superior já que tipicamente são aceitas de 8 a 10. Isso se deve à qualidade das ferramentas submetidas, todas com excelentes contribuições para a comunidade de Redes de Computadores e Sistemas Distribuídos. Os principais tópicos deste ano são Redes Definidas por Software, Redes sem-fio e IoT e Visualização e Extração de Dados. No total, teremos 4 Sessões Técnicas.

Este ano o Salão de Ferramentas será aberto por uma palestra cujo título é "Bockchain & IoT: Desafios tecnológicos para um dos maiores mercados de tecnologia do futuro!" A expectativa é que autores e participantes do Salão assim como do SBRC possam assistir a palestra e discutir futuros aspectos e oportunidades neste tema tão interessante atualmente.

Por fim, agradeço a todos os membros do Comitê de Programa que fizeram excelentes revisões contribuindo para termos um Salão de Ferramentas de alta qualidade. Agradeço aos autores que submeteram seus trabalhos e pela disposição em apresentá-los durante o SBRC. Agradeço também ao Daniel Batista, organizador do Salão de Ferramentas de 2016, pelas conversas e explicações fornecidas. Finalmente, agradeço aos organizadores do SBRC 2017, Eduardo Cerqueira e Antônio Abelém, pela confiança em mim depositada e pelo suporte fornecido.

Nos vemos em Belém!

Fábio Luciano Verdi

Coordenador do Salão de Ferramentas do SBRC 2017

Comitê de Programa da Trilha Principal

- Alberto Schaeffer-Filho (UFRGS)
- Aldri dos Santos (UFPR)
- Alex Vieira (UFJF)
- Alfredo Goldman (IME – USP)
- Allan Freitas (IFBA)
- Alysson Bessani (Universidade de Lisboa)
- Ana Paula Couto da Silva (UFMG)
- André Drummond (UnB)
- Andre Aquino (UFAL)
- Andrey Brito (UFCG)
- Anelise Munaretto (UTFPR)
- Antônio Abelém (UFPA)
- Antonio Alfredo Ferreira Loureiro (UFMG)
- Antonio Rocha (IC/UFF)
- Antonio Tadeu Gomes (LNCC)
- Artur Ziviani (LNCC)
- Bruno Schulze (LNCC)
- Carlos Ferraz (UFPE)
- Carlos Kamienski (UFABC)
- Carlos Alberto Vieira Campos (UNIRIO)
- Carlos Mauricio Figueiredo (UEA)
- Célio Vinicius Neves de Albuquerque (UFF)
- Cesar Marcondes (UFSCAR)
- Christian Esteve Rothenberg (UNICAMP)
- Cristiano Both (UFCSPA)
- Daniel Batista (IME – USP)
- Daniel Fernandes Macedo (UFMG)
- Daniel Guidoni (UFSJ)
- Daniel Menasche (UFRJ)
- Dênio Mariz Sousa (IFPB)
- Divanilson Campelo (UFPE)
- Djamel Fawzi Hadj Sadok (UFPE)
- Dorgival Guedes (UFMG)
- Eduardo Cerqueira (UFPA)
- Eduardo Nakamura (UFAM)
- Elias Duarte Jr. (UFPR)
- Fabíola Greve (UFBA)
- Fatima Duarte-Figueiredo (PUC-Minas)
- Fábio Luciano Verdi (UFSCAR)
- Fernando Dotti (PUC-RS)
- Flavia Delicato (UFRJ)
- Francisco Brasileiro (UFCG)

- Geraldo Robson Mateus (UFMG)
- Gustavo Figueiredo (UFBA)
- Gustavo Pavani (UFABC)
- Heitor Ramos (UFAL)
- Hermes Senger (UFSCAR)
- Horàcio Oliveira (UFAM)
- Humberto Marques (PUC-Minas)
- Igor Moraes (UFF)
- Islene Garcia (UNICAMP)
- Italo Cunha (UFMG)
- Jacir Bordim (UnB)
- Jó Ueyama (USP)
- Joni da Silva Fraga (UFSC)
- José de Souza (UFC)
- José Ferreira de Rezende (UFRJ)
- José Augusto Suruagy Monteiro (UFPE)
- Jose-Marcos Nogueira (UFMG)
- Jussara Almeida (DCC-UFMG)
- Kelvin Dias (UFPE)
- Lásaro Jonas Camargos (UFU)
- Lau Cheuk Lung (UFSC)
- Leandro Villas (UNICAMP)
- Leobino Sampaio (UFBA)
- Liane Margarida Rockenbach Tarouco (UFRGS)
- Linnyer Ruiz (UEM)
- Lisandro Zambenedetti Granville (UFRGS)
- Luci Pirmez (UFRJ)
- Luciano Paschoal Gaspary (UFRGS)
- Luis Carlos De Bona (UFPR)
- Luis Henrique Costa (UFRJ)
- Luiz Eduardo Buzato (UNICAMP)
- Luiz Fernando Bittencourt (UNICAMP)
- Luiz Filipe Vieira (UFMG)
- Luiz Henrique Correia (UFLA)
- Magnos Martinello (UFES)
- Marcel William Rocha da Silva (UFRRJ)
- Marcelo Dias de Amorim (LIP6/CNRS – UPMC, França)
- Marcelo Rubinstein (UERJ)
- Marcos Vieira (UFMG)
- Marinho Barcellos (UFRGS)
- Markus Endler (PUC-Rio)
- Mauro Fonseca (UTFPR)
- Michael Stanton (RNP & UFF)
- Michelle Wangham (Univali)
- Miguel Elias Mitre Campista (UFRJ)

- Moises Ribeiro (UFES)
- Nabor Mendonca (UNIFOR)
- Natalia Castro Fernandes (UFF)
- Nazareno Andrade (UFCG)
- Nelson Luis Saldanha da Fonseca (UNICAMP)
- Noemi Rodriguez (PUC-Rio)
- Olga Goussevskaia (UFMG)
- Otto Carlos Muniz Bandeira Duarte (UFRJ)
- Paulo Aguiar (UFRJ)
- Paulo Cunha (UFPE)
- Paulo Pires (UFRJ)
- Paulo André da Silva Gonçalves (UFPE)
- Pedro Velloso (UFRJ)
- Pedro Olmo Vaz de Melo (UFMG)
- Rafael Esteves (IFRS)
- Rafael Pasquini (UFU)
- Raimundo Jose de Araujo Macedo (UFBA)
- Raquel Lopes (UFCG)
- Raquel Mini (PUC-MG)
- Reinaldo Braga (IFCE)
- Ricardo R. Oliveira (UFOP)
- Rodolfo Villaca (UFES)
- Rodrigo de Souza Couto (UERJ)
- Rodrigo Fonseca (Brown University)
- Ronaldo Ferreira (UFMS)
- Ronaldo Salles (IME)
- Rossana Andrade (UFC)
- Sidney Lucena (UNIRIO)
- Silvana Rossetto (UFRJ)
- Stenio Fernandes (UFPE)
- Thais Vasconcelos Batista (UFRN)
- Wagner Meira Jr. (UFMG)
- Weverton Cordeiro (UFRGS)
- William Giozza (UnB)

Revisores da Trilha Principal

- Adriano Branco
- Alberto Schaeffer-Filho
- Alberto Sampaio Lima
- Aldri dos Santos
- Alex Borges Vieira
- Alextian Liberato
- Alfredo Goldman
- Alice Menezes
- Allan Freitas
- Aloizio Silva
- Alysson Bessani
- Américo Sampaio
- Ana Paula Couto da Silva
- Anílton Salles Garcia
- Anderson Fernandes Pereira dos Santos
- André Drummond
- André Lage
- André Luiz Nasserala Pires
- Andre Aquino
- Andressa Vergutz
- Andrey Brito
- Anelise Munaretto
- Antônio Abelém
- Antonio Alfredo Ferreira Loureiro
- Antonio Lobato
- Antonio Rocha
- Antonio Augusto Ribeiro Coutinho
- Antônio Tadeu Azevedo Gomes
- Arthur Jacobs
- Artur Ziviani
- Atslands Rocha
- Benevid Felix
- Bruno Chang
- Bruno Costa
- Bruno Kimura
- Bruno Schulze
- Carina Teixeira de Oliveira
- Carlos Castellani
- Carlos Ferraz
- Carlos Kamienski
- Carlos Alberto Vieira Campos
- Carlos Mauricio Figueiredo
- Carlos Renato Storck

- Célio Vinicius Neves de Albuquerque
- Cesar Marcondes
- Christian Esteve Rothenberg
- Claiton Soares
- Claudio de Farias
- Cristiano Both
- Cristiano Silva
- Cristina Dominicini
- Dalbert Mascarenhas
- Daniel Batista
- Daniel Cason
- Daniel Dias
- Daniel Fernandes Macedo
- Daniel Guidoni
- Daniel Marcon
- Daniel Menasche
- Danielle Ferreira
- Danilo Possati
- Davi Böger
- Dênio Mariz Sousa
- Dianne Medeiros
- Diego Rossi Mafioletti
- Diogo Mattos
- Divanilson Campelo
- Djamel Fawzi Hadj Sadok
- Dorgival Guedes
- Eder John Scheid
- Edjair Mota
- Eduardo Alchieri
- Eduardo Cerqueira
- Eduardo Feitosa
- Eduardo Nakamura
- Efren Souza
- Elias Duarte Jr.
- Emanuel Coutinho
- Eros Spalla
- Everton Cavalcante
- Fabíola Greve
- Fabio Morais
- Fabio Verdi
- Fabrício Silva
- Fatima Duarte-Figueiredo
- Felipe Domingos da Cunha
- Fernando Dotti
- Flavia Delicato

- Flávio Assis Silva
- Francisco Brasileiro
- Georges Daniel Amvame-Nze
- Geraldo Pereira
- Geraldo Robson Mateus
- Giacomo Mc Evoy
- Gilmar Vassoler
- Glederson Santos
- Guilherme Luiz Moritz
- Gustavo Figueiredo
- Gustavo Maciel Dias Vieira
- Gustavo Pavani
- Heitor Freitas
- Heitor Ramos
- Helga Balbi
- Henrique Moura
- Hermes Del Monego
- Hermes Senger
- Horácio Oliveira
- Hugo de Freitas Siqueira Sadok Menna Barreto
- Humberto Marques
- Ian Bastos
- Igor Dos Santos
- Igor Moraes
- Igor Jochem Sanz
- Islene Garcia
- Italo Cunha
- Ivanilton Polato
- Jacir Bordim
- Jair Leite
- Jó Ueyama
- João Pinto Neto
- Joelias Junior
- Joni da Silva Fraga
- José De Souza
- José Ferreira de Rezende
- José Augusto Suruagy Monteiro
- Jose Torres Neto
- Jose-Marcos Nogueira
- Joubert Lima
- Juliana de Santi
- Juliano Naves
- Jussara Almeida
- Kelvin Dias
- Klaus Wehmuth

- Lasaro Camargos
- Leandro de Sales
- Leandro Resendo
- Leandro Villas
- Leandro Yukio Mano Alves
- Leobino Nascimento Sampaio
- Leônidas Lima Junior
- Leonardo Bays
- Leonardo Botega
- Leonardo Oliveira
- Liane Margarida Rockenbach Tarouco
- Linnyer Ruiz
- Lucas Bondan
- Luci Pirmez
- Luciano Barreto
- Luciano Paschoal Gaspary
- Luis Faina
- Luis Lima
- Luis Carlos De Bona
- Luis Henrique Costa
- Luiz Eduardo Buzato
- Luiz Fernando Bittencourt
- Luiz Filipe Vieira
- Luiz Henrique Andrade Correia
- Magnos Martinello
- Marcel William Rocha da Silva
- Marcelo Alves
- Marcelo Caggiani Luizelli
- Marcelo Dias de Amorim
- Marcelo Pellenz
- Marcelo Rubinstein
- Marcelo Santos
- Marcia Helena Moreira Paiva
- Marcio Ferro
- Marcio Maia
- Marcos Vieira
- Marcus Carvalho
- Marinho Barcellos
- Mariza Ferro
- Markus Endler
- Martin Andreoni Lopez
- Mateus Pelloso
- Matheus Lehmann
- Mauro Fonseca
- Maxwell Monteiro

- Maycon Leone
- Maykon de Souza
- Márcio Carvalho
- Michael Stanton
- Michel Bonfim
- Michelle Wangham
- Miguel Neves
- Miguel Elias Mitre Campista
- Moises Ribeiro
- Muriel Franco
- Nabor Mendonca
- Natalia Castro Fernandes
- Nazareno Andrade
- Nelson Fonseca
- Nivia Cruz Quental
- Noemi Rodriguez
- Odorico Mendizabal
- Olga Goussevskaia
- Paulo Aguiar
- Paulo Cunha
- Paulo Mafra
- Paulo Pires
- Paulo Rego
- Paulo André da Silva Gonçalves
- Pedro de Botelho Marcos
- Pedro Velloso
- Pedro Henrique Cruz Caminha
- Pedro Olmo Vaz de Melo
- Petronio Júnior
- Rafael da Silva
- Rafael de Sousa Junior
- Rafael Esteves
- Rafael Guimarães
- Rafael Pasquini
- Rafael Souza
- Raimundo Jose de Araujo Macedo
- Raquel Lopes
- Raquel Mini
- Raquel S. Cabral
- Reinaldo Braga
- Renê Oliveira
- Ricardo Pfitscher
- Ricardo R. Oliveira
- Ricardo Rios
- Ricardo Luis dos Santos

- Roberto Yokoyama
- Roberto Iraja Tavares da Costa Filho
- Rodolfo Villaca
- Rodrigo de Souza Couto
- Rodrigo Fonseca
- Rodrigo Ruas Oliveira
- Ronaldo Ferreira
- Ronaldo Salles
- Roni Shigueta
- Rossana Andrade
- Sidney Lucena
- Silvana Rossetto
- Stenio Fernandes
- Tamer Cavalcante
- Taniro Rodrigues
- Thais Braga
- Thais Vasconcelos Batista
- Tiago Ferreto
- Vinícius Schaurich
- Vinicius Mota
- Wagner Meira Jr.
- Wendley Silva
- Weverton Cordeiro
- William Giozza

Comitê de Programa do Salão de Ferramentas

- Alfredo Goldman (IME – USP)
- André Drummond (UnB)
- Antonio Rocha (IC/UFF)
- Antonio Marcos Alberti (INATEL)
- Bruno Schulze (LNCC)
- Carlos Kamienski (UFABC)
- Cesar Marcondes (UFSCar)
- Cintia Margi (USP)
- Daniel Batista (IME – USP)
- Daniel Cordeiro (USP)
- Divanilson Campelo (UFPE)
- Eduardo Cerqueira (UFPA)
- Fabio Verdi (UFSCar)
- Flavio Silva (UFU)
- Gustavo Figueiredo (UFBA)
- Hélio Guardia (UFSCar)
- Hermes Senger (UFSCar)
- Iguatemi E. Fonseca (UFPB)
- Kalinka Castelo Branco (ICMC – USP)
- Leandro Villas (UNICAMP)
- Leobino Sampaio (UFBA)
- Lisandro Zambenedetti Granville (UFRGS)
- Luciano de Paula (IFSP)
- Luiz Theodoro (Algar Telecom)
- Luiz Fernando Bittencourt (UNICAMP)
- Nelson Rosa (UFPE)
- Olga Goussevskaia (UFMG)
- Pedro Frosi-Rosa (UFU)
- Rafael Pasquini (UFU)
- Raphael Camargo (UFABC)
- Rodolfo Villaca (UFES)
- Ronaldo Ferreira (UFMS)
- Tarcisio Rocha (UFS)
- Yeda Regina Venturini (UFSCar)

Sumário

Trilha Principal do SBRC 2017

Salão de Ferramentas 2017

Trilha Principal do SBRC 2017
Sessão Técnica 1
Computação nas Nuvens I

BACOS: A Dynamic Load Balancing Strategy for Cloud Object Storage

Manoel Rui P. Paula[1], **Eduardo Rodrigues**[1],
Victor A. E. Farias[1], **Flávio R. C. Sousa**[1], **Javam C. Machado**[1]

[1]LSBD – Departamento de Computação – Universidade Federal do Ceará (UFC)
Campus do Pici – Bloco 952 – 60.020-181 – Fortaleza – CE – Brazil

{manoel.rui,eduardo.rodrigues}@lsbd.ufc.br
{victor.farias,flavio.Sousa,javam.machado}@lsbd.ufc.br

Abstract. *Cloud Computing is an efficient model for processing and storing large amounts of data. The cloud is composed by heterogeneous resources and has a variable workload. Cloud object storage systems arise as an efficient manager for data using heterogeneous devices, regarding storage capacity and performance. In the cloud, since workload changes dynamically, the dynamic reconfiguration is needed to improve resource utilization. Thus, load balancing techniques are crucial to redistribute workload among the processing nodes to avoid underloading or overloading. Conventional load balancing strategies are only aware of storage devices' capacity, resulting in system performance degradation. To address these limitations, this paper presents a non-intrusive approach to load balancing in the cloud which considers storage devices with heterogeneous performance. Experimental results confirm that our approach improves performance in terms of response time and throughput when compared to the strategy employed by Openstack Swift object storage.*

1. Introduction

Cloud computing is a paradigm of remarkable success for service-oriented computing. Modern data-driven applications in the cloud demand large computing resources. In this scenario, distributed data storages are crucial components on the software stack. For this purpose, cloud object storage has emerged to support such high requirements [Mesnier et al. 2003]. Cloud object storage systems have scalable architecture and are composed by distributed nodes responsible for storing and retrieving data distributed among several servers and their storage devices. The communication among those nodes is typically through a high performance network. These object storage systems provide a high level interface to abstract low level layers of storage devices such as local file system. This high level layer is commonly used to read and write unstructured data as objects, often being multimedia data like documents, images, videos, audio etc. [Mesnier et al. 2003]

Commodity storage devices are used on cloud object storage systems to store data. Therefore, it is possible to extend the total storage space capacity with reduced cost. Most of commodity storage devices are not reliable, but data durability is achieved using replication mechanisms that can be provided by a cloud object storage. However, those devices may be a bottleneck because of their poor performance. [Gunawi et al. 2005] In order to improve system's throughput and latency in storage layer, providers offer storage services

with high performance storage devices as Solid State Drive (SSD) and Serial Attached SCSI (SAS) hard drives. In general, high performance storage devices are composed by costly components or are difficult to be produced, for this reason, their storage capacity is reduced and they are expensive. Consequently, it is infeasible to maintain a storage system where most of the storage devices are high performance and it is interesting to not overload them, avoiding frequent replacements.

Commercial object storage systems as Ceph [Weil et al. 2006], OpenStack Swift [OpenStack 2017], and GlusterFS [GlusterFS 2017] are mainly designed to store large amounts of data on a pool of storage devices based only on the device capacity while not focusing on the device performance aspect. So, they fail in leveraging performance of these devices, especially for devices with distinct (heterogeneous) performance.

In cloud environment, load imbalance happens when a cloud object store dynamically handle different types of operations like read, write and delete from several clients. Eventually, some system components can not manage requests due to lack of resources in that moment, hence compromising system performance [Deshmukh and Deshmukh 2015]. It is a non-trivial problem because is hard to know future system load to adjust the current system resource utilization, especially in a heterogeneous system containing some device components faster then others. Works in state-of-art such as [Tan et al. 2013] and [Hsiao et al. 2013] try to solve this problem rebalancing current storage system load through a data migration scheme between system components to readjust underloaded and overloaded resources. In this scenario, load balancing policies arise as an effective strategy to tackle the performance aspect of object storage systems and improving utilization of heterogeneous devices.

In this paper, we address the load balancing problem to cloud object storage. The term *balance* and *rebalance* are interchangeable in this paper. We introduce BACOS: a dynamic load BAlancing strategy to Cloud Object Storage. BACOS uses previous knowledge about the workload behavior and handle distinct storage device performance to improve system's overall performance. Specifically, we can highlight the following contributions:

- Non-intrusive model that handle storage devices with heterogeneous performance for distributed object storage systems.
- A strategy to adapt to workload changes.
- Testbed for testing production-ready systems. Specifically, we tested with a commercial object storage system.

Organization: The remaining of this paper is organized as follows. Section 2 surveys related work. Section 3 gives a brief introduction on the architecture of an object storage system suitable to BACOS. Section 4 explains BACOS strategy. The implementation and experimental evaluation of BACOS is presented in Section 5. Finally, Section 6 presents the conclusions and future works.

2. Related Work

The work presented in [Wang et al. 2015] proposes a lightweight framework of workload balancing and adaptive resource management for Swift object storage. The framework was designed to balance the workload regularizing virtual and physical nodes. To relieve

the overloaded resources in the distributed physical nodes the framework, instead of migrating data between the nodes, migrates virtual machines between physical machines while the system is running. Additionally, the approach is not intrusive and is designed to be used in a virtual environment through a well-known API. However, how to configure the values of parameters based on a workload analysis is not explored. Also, the authors didn't include storage capabilities to their model.

The work presented in [Chung et al. 2012, Hsiao et al. 2013] proposes a fully distributed load rebalancing algorithm where each storage node is responsible for balance its own load spontaneously, eliminating the need for a central coordinator. The approach implements a DHT protocol to get a fast lookup and handle the load imbalance problem by migrating data load from the heavier nodes to the lighter ones, so that after few interactions all file chunks would be as uniformly distributed as possible among all storage nodes. Additionally, the proposed algorithm aims to reduce network traffic caused by the rebalancing process as much as possible, although it does not take advantage of performance qualities from the storage backend since the main objective is balance the data among storage servers in regard to their storage capacity.

The Adaptive Loading Data Migration in Distributed File System (ALDM) [Tan et al. 2013] strategy is a dynamic algorithm to balance the load of distributed file systems using data migration to mitigate the effect of frequent access to popular data. This approach proposes a measure mechanism to represent the system load status by pricing the system resources against their degradation along time, like network, disk I/O and disk capacity. A central controller node is responsible to dynamically decide when to execute the data migration, which files will be migrated, and where to migrate, to minimize the migration costs. The experiments show that the proposed algorithm can effectively balance the load of data servers, increasing the bandwidth after the migration process. Furthermore, ALDM seems a good adaptability approach to various loads, although it has been tested only in an environment with homogeneous resources.

3. Object Storage Architecture

In this work we assume a cloud object storage architecture as depicted in Figure 1. A cloud object storage is composed of one or more proxy nodes and a set of storage nodes. A storage node is a server that maintains at least a storage device such as a HDD or SSD, which is able to store objects in non-volatile memory. A proxy node is responsible for receiving requests from clients and redirect them to storage nodes.

A Distributed Hash Table (DHT) is a distributed system that provides the functionality of a hash table, mapping a key k to a node n [Felber et al. 2014]. The implementation of DHT is crucial to a object storage performance since the dispersion of the keys among nodes defines the proportion of objects in a storage device. They can improve system performance using a variant of DHT called consistent hash. Where a consistent hash is a hash function such that when a hash table is resized, only K/N keys need to be remapped on average, where K is the number of keys, and N is the number of nodes. An object storage system can also support different types of storage policy, for example, erasure code [Li and Li 2013] or partial replication policy [Nuaimi et al. 2013]. Examples of commercial cloud objects storages are Ceph [Weil et al. 2006], OpenStack Swift [OpenStack 2017], and GlusterFS [GlusterFS 2017].

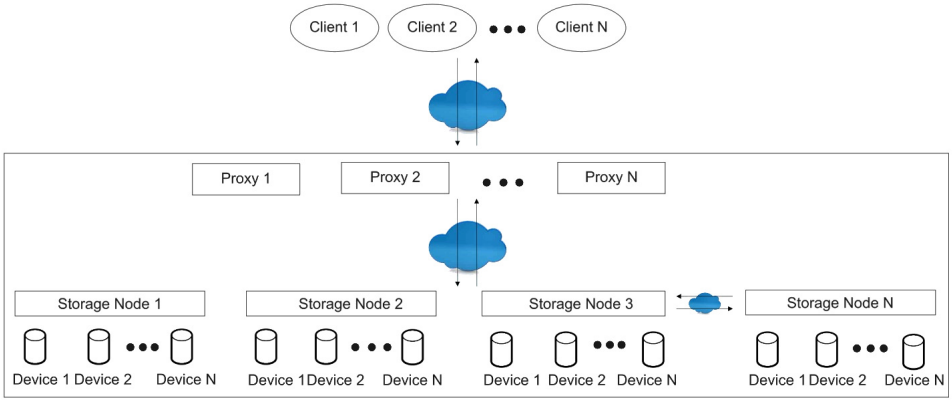


Figure 1. Object Storage Architecture.

In this work, we focus on Openstack Swift, which is an open source object storage system widely used in production by many companies, e.g. Rackspace [Rackspace 2017] and SwiftStack [SwiftStack 2017]. Swift ensures the data is replicated across the cluster to increase data availability and durability. The location where the data should reside in the cluster is determined by a ring, which is responsible to map the data and its replica from the logical locations to their physical locations. The Swift's ring uses a consistent hashing. During the creation of the ring, which is not an automated process, it is possible to set up values like replication count and weight for each device in the cluster. The weight of each device determines the proportion of keys over its partitions. Thus, it causes a direct impact in the distribution of the data among the storage nodes. Note that the data distribution inducts the proportion of requests toward devices.

Swift architecture is composed by different web service nodes and background processes where each component is scalable. The web service nodes are classified in two main categories: proxy server type and storage node type (composed by an account server, a container server and an object server). Each proxy server node and storage node contain a copy of the ring. The background processes are responsible for data replication, data reconstruction, data updating and data auditing. Read and write requests from/to Swift obey the following rules: for reading requests, only one replica node is enough to retrieve the object and return success to the client; for writing requests, the quorum of half of total replica nodes plus one are needed to persist the object and return success to the client.

4. The BACOS Approach

The main purpose of our strategy is to balance requests toward an object storage system by redistributing data proportionally to the performance of the storage devices. Moreover, it redistributes data dynamically according to the workload changes to ensure that the storage system operates in optimal performance. BACOS is based on previous knowledge obtained by a set of offline experiments in a test environment isolated from production system.

Figure 2 shows BACOS's load balancing flow. The storage device classification step measure the performance of devices, identify devices with similar performance and assign them to performance classes; In the system performance measurement step, several workloads are issued to the storage system to measure system performance results

under distinct data distributions among storage devices. In last step, the load balancing method uses a lookup table to find the best data distribution that improves the storage system performance while avoiding overload faster storage devices and minimizing data migration.

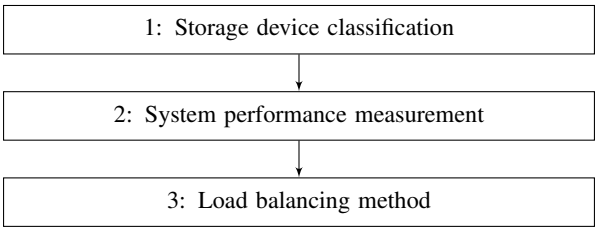


Figure 2. BACOS's load balancing flow.

4.1. Storage Device Classification

To propose a storage device classification, we define a storage device as $d_k \in \{d_1 \dots d_Z\}$, where Z is the total number of storage devices in storage system. Furthermore, we assume the performance of d_k as the max number of Input/Output Operations Per Seconds (IOPS). Then, a benchmark tool is used to issue read and write operations toward the device and is measured the resultant IOPS in order to discover the supported IOPS by a storage device. We recommend using 50% for read and write operations and $1GB$ file for benchmark parameters. This process is done only when a new storage device is attached to storage system.

Once discovered the IOPS of each device, the devices are grouped into performance classes C_j, for $1 \leq j \leq L$, such that devices belonging to the same class have similar performance. As we are dealing with heterogeneous device performance, at least two performance classes should exist, where L is the total number of classes. These classes are constructed by an user-defined threshold for IOPS T_{C_j} per class, that defines which performance class a device is assigned. Assume, without any loss, that $T_{C_1} < T_{C_2} < \cdots < T_{C_L}$. Thus a device d_k is assigned to class C_j where j is the largest $j \in \{1, 2, \cdots, L\}$ such that the IOPS of d_k is larger than T_{C_j}. Note that by the construction of the performance classes, devices in class C_i are faster than devices in class C_j if $1 \leq j < i \leq L$. These performance classes are useful for simplifying the attributes of heterogeneous devices in the storage system and reducing the management complexity of number of devices in the next steps of our approach.

4.2. System Performance Measurement

In this step, we execute several experiments in order to understand the behavior of the storage system. These experiments are run in a test environment isolated from the production environment. In each experiment, we vary workload parameters and system parameters. However, it is important to highlight that BACOS uses high level interfaces from storage system to setup the proportion of data and requests that a device should manage as a system parameter. For this reason, we define class weight $W = (w_1, w_2, \cdots, w_L)$, where the weight w_j specifies the proportion of data and requests that a performance class C_j manage. Hence, w_j defines the proportion of data and requests that a device manage in a performance class. Therefore, in an experiment, we use only the class weight W as

system parameter. The workload parameters that we employ are number of clients N and read/write ratio RW. Thereby, we assign a list of values for each parameter, shown in section 5.2. For each parameter combination (N, RW, W), we execute an experiment of 5 minutes with 30 seconds of warm up and 30 seconds of cool down. Thus we collect and record the mean response time RT, and the total number of errors N_{error} that reflects bad resource utilization or lack of resources by storage system.

4.3. Load Balancing Method

In this step, BACOS generates a lookup table to hold the optimal W for a given workload configuration that maximize the system performance. The optimal class weight is defined by W_{opt}. The lookup table is based on the experiments done in a previous step. In production environment, BACOS monitor the current workload and consult the table to get the W_{opt} for the current moment.

We divide our load balancing method in two substeps: i) the first substep in section 4.3.1, criteria for optimal class weight, presents a score function F as main criterion to evaluate the W_{opt} for each workload configuration and ii) the second substep in section 4.3.2, lookup table construction, shows how the lookup table is created and how to use it in a production system.

4.3.1. Criteria for optimal weights

Since we aim to construct a lookup table in the next substep (section 4.3.2), we use metrics collected in step 2 (section 4.2) to choose an optimal class weight W_{opt} that improves system performance without overloading storage systems. Then, we define a cost function F in Equation 1 which scores the performance of an experiment given its parameters (N, RW, W) with response time denoted by RT and the total error number denoted by N_{error}. Consider $Error_{mean}$ the mean number of error, RT_{max} the maximum mean response time and RT_{min} the minimum mean response time of all experiments executed using a specific workload configuration (N, RW) with the set of class weights in list L_w.

$$F = \frac{RT + (RT_{min} \times P)}{RT_{max} + RT_{min}} \tag{1}$$

Where P is a penalty function for errors in the experiments as depicted in Equation 2. Function P penalizes experiments in which a large number errors occurred. These errors are failed operations, which indicate unbalancing in the storage system, producing overloaded or underloaded storage devices.

$$P = \frac{N_{error}}{N_{error} + Error_{mean}} \tag{2}$$

Note that F is normalized by RT_{max} to generate a final score value between 0 and 1. The smallest the value of F, the best is the performance of the experiment.

4.3.2. Lookup Table Construction

In order to construct the lookup table T, it is necessary to find the optimal class weights W_{opt} that maximize the performance of the system for each workload configuration. In step 2 (section 4.2), for each tested workload configuration, i.e., a pair (N, RW), we did many experiments with a set of class weights W in list L_w. To choose the W_{opt} for a given pair (N, RW), we pick up the best experimented $W \in L_w$ in terms of performance that minimized function F. In the end of process, the lookup table mapped a pair (N_i, RW_j) to a W_{opt_k}. Once the table T is created, BACOS load balance method is ready to query T to get W_{opt} even to a unknown N and RW. BACOS use the nearest neighbor interpolation method [Lehmann et al. 1999] in T to discover the desirable W_{opt_k} from (N_i, RW_j).

```
   input : T ← {(N_1, RW_1) : W_opt_1, ..., (N_i, RW_j) : W_opt_k}
           Dev ← {C_1 : [d_1, ..., d_Z], ..., C_L : [d_1, ..., d_Z]}
           α
   output: 0: no need for load balancing
           1: load balancing success
 1 N^t ← collectCurrentNumberOfClients();
 2 RW^t ← collectCurrentReadWriteRatio();
 3 W^t ← collectCurrentClassWeight();
 4 W_opt ← T[(N^t, RW^t)];
 5 W^t ← W^t + α × (W^t − W_opt);
 6 if dataMigration() or W^t ≈ W_opt then
 7 |   return 0;
 8 else
 9 |   while W^t ≠ W_opt do
10 |   |   for l ← 1 to L do
11 |   |   |   deviceList ← Dev[C_l];
12 |   |   |   foreach dev in deviceList do
13 |   |   |   |   w_dev ← getDeviceWeight(W^t, len(deviceList));
14 |   |   |   |   apply w_dev to dev;
15 |   |   |   end
16 |   |   end
17 |   |   W^t ← W^t + α × (W^t − W_opt);
18 |   end
19 |   return 1
20 end
```

Algorithm 1: BACOS load Balance Algorithm.

The Algorithm 1 presents a pseudo code of BACOS in production system. The first input is the lookup table T containing pairs (N_i, RW_j) as access key and W_{opt_k} as result value. The second input Dev is a dictionary containing a set of storage devices per performance class derived from first step of our strategy (section 4.1). The last input α is the transfer migration factor that represents the proportion of the whole data intended to be migrated per interaction, where $0 \leq \alpha \leq 1$. This parameter is intended to avoid a system collapse during the migration process while new requests continue to arrive. To choose the α value, a trade-off between fast convergence regarding the number of interactions and system performance during object migration process must be considered. The α parameter should be computed based on utilized storage system capacity to mitigate the problem of transferring a huge volume of data through the network and overload storage

system resources. Parameter estimation is out of scope of this work. On lines 1 to 3, current workload and system parameters in time t are collected. On lines 4 to 5, W_{opt} from T is acquired and the new class weight W^t from α are estimated to reach W_{opt}. On lines 6 and 7, it is verified if there are any data migrations in execution or if W^t is very close to the expected W_{opt} at the current moment. In both cases, there is no need to do load balancing. On lines 9 to 19, the object migration between storage devices is done, starting from current W^t to W_{opt}. When W_t is close enough to W_{opt}, the load balance process converge and is finished with success. On lines 13 and 14 the individual proportion of data that each storage device should manage is computed, denoted by weight w_{dev}, and the data proportion of a performance class is spread across all devices in the class.

5. Performance Evaluation

In this section, we describe the experiments used to evaluate the performance of our strategy. The main goal is to show that our strategy can improve storage system performance in a environment with storage devices with heterogeneous performance. We measure performance in terms of three metrics: response time, throughput, and success rate. These experiments are conducted in a private cloud using Openstack Swift Object Storage [OpenStack 2017]. We compare our approach to a baseline strategy recommended in Swift documentation where is suggested to set static weights proportionally to the storage device capacity. For instance, two devices with storage capacity of 1 TB and 2 TB would have weights set as 1000 and 2000, respectively.

The chosen evaluation method was the confidence interval of the mean of the difference between two performance distributions [Jain 1991]. To assess this confidence interval, the samples from the two distributions (BACOS and Swift default strategy) are paired, as they are collected under the same conditions, and their difference results in another distribution. If the confidence interval includes zero, the performance distributions are not significantly different. Otherwise, by analyzing the values of the resulting distribution, it is possible to know how much one strategy has improved the other. To build the confidence interval, we consider a Student's t-distribution with 95% confidence level. This procedure was applied to read and write operations in response time and throughput.

5.1. Environment

To measure the performance of our approach, we used Cloud Object Storage Benchmark (COSBench) tool version 0.4.2 rc2 [Zheng et al. 2013]. COSBench has supported many cloud object storage solutions on the market like Swift, Amazon S3, and Ceph thus making easier any future comparison among those cloud object storage solutions.

We deployed and tested a BACOS prototype in a private cloud on Openstack environment configuring the ring DHT with 2 object replicas. All virtual machines in our experiments were provided by Openstack Nova and storage devices by Openstack Cinder. The virtual machine hosting COSBench was a 4 vCPUs, 8 GB main memory and 80 GB storage capacity. The proxy node configuration had a VM instance with 16 vCPUs, 16 GB main memory and 160 GB storage capacity. We deployed 2 storage nodes, $sn1$ and $sn2$, each one provided with 3 storage devices and with 4 vCPUs, 8 GB main memory and 80 GB storage capacity. The storage node $sn1$ had heterogeneous devices, one with 10 GB and 100 IOPS, another device with 20 GB and 500 IOPS and the last one with 40

GB and 1000 IOPS. The storage node $sn2$ was provided with homogeneous devices in terms of storage capacity and performance, where each one had a device with 40 GB and 100 IOPS.

5.2. Evaluation scenario

The evaluation scenario consists in issuing a workload from COSBench to object storage Swift and collecting metrics like response time, throughput and success ratio to further analysis. We created 32 containers with 5000 objects per container resulting a total of 160000 objects in our experiments, where 2500 objects per container were reserved to read operations and another 2500 objects to write operations. We inserted half of the total number of objects in Swift to simulate an object storage system already in production and required to start concurrent read and write operations. The objects are selected following a uniform distribution. From now on, we will refer to the approach suggested by Swift's documentation, which consists of statically configuring the weights of storage devices proportionally to the device storage capacity [OpenStack 2017] as the baseline approach.

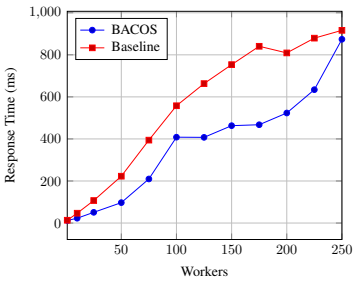
The workload and system parameter values for system performance measurement step (section 4.2) of our approach are: The number of clients N or workers to COSBench tool takes values in $[10, 50, 100, 150, 200, 250]$; we fixed the read/write ratio at $80/20$ and 4KB object size; we define empirically the parameter $\alpha = 1$ for our environment; $W = (W_1, W_2, W_3)$ as a three-sized tuple for weights of 3 performance classes containing integers W_1, W_2, W_3, where we vary W_i from 0 to 1000 with step 50, $W_1 + W_2 + W_3 = 1000$ and $W_1 \leq W_2 \leq W_3$. We used similar parameters from performance measurement step to evaluation, however, the difference when N assumes values in $[1, 10, 25, 50, 100, 125, 150, 175, 200, 225, 250]$ to compare BACOS and baseline are in different scales. The resulting performance metrics (response time and throughput) are collected every second during 300s of observation time for a given N, only discarding 30s from the start and 30s from the end to avoid noises. The collected performance metric values for a given N were aggregated in a mean and the set of mean values of each N represents a distribution of a performance metric.

5.2.1. Response Time

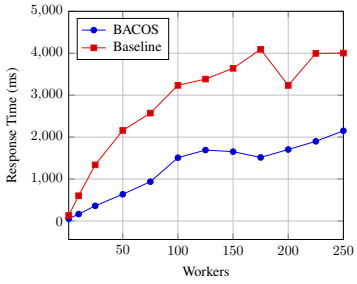
Regarding the response time for read and write operations, BACOS provided low response time values compared to the baseline as depicted in Figure 3. We computed 95% of confidence interval for the mean of difference of response time distributions between the baseline and BACOS for read and write operations as depicted in Figure 4. Thus, we can conclude that our approach was better than the baseline since the zero value was not included in the interval and positive sign of mean denoted that the baseline response times were higher than ours. The Figures 3(a) and 4(a) show the comparison of the response time for read operations between our approach and the baseline. The response time differences between the two approaches increases as the number of workers increases too, since few read operations underutilize storage device resources. Although there is a clear difference between performance distributions of both strategies for read response time, this difference could be greater because of the default replication choice approach adopted by Openstack Swift and considered for both strategies in evaluation. The replica choice approach chooses a replica with same probability between storage nodes and its

decision is not aware of storage device performance. Even so, since in our strategy the largest proportion of replicas are stored in the faster storage devices, choosing a replica from these devices makes the read operation faster than the baseline, where the proportion of replicas are based in the storage capacity and not in the performance.

The same behavior of read response times happens with write response times depicted in Figures 3(b) and 4(b) when the number of workers increases. However, the large difference in response time values between our strategy and baseline in write operations was due to the quorum decision of Openstack Swift. As we worked with two replicas per object, just one replica is necessary to confirm the successful operation. Consequently, there is a high probability that the first replica of an object has confirmed by a fast storage device.

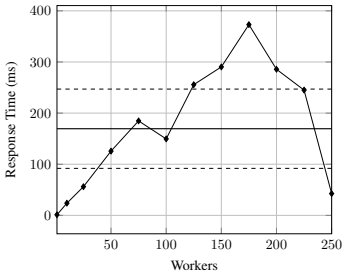


(a) Read response time.

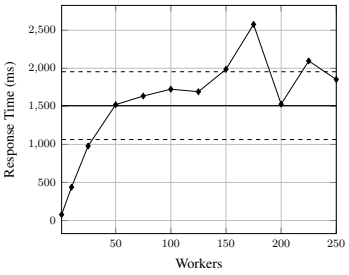


(b) Write response time.

Figure 3. Response time distributions.



(a) Difference of read response time between baseline and BACOS.



(b) Difference of write response time between baseline and BACOS.

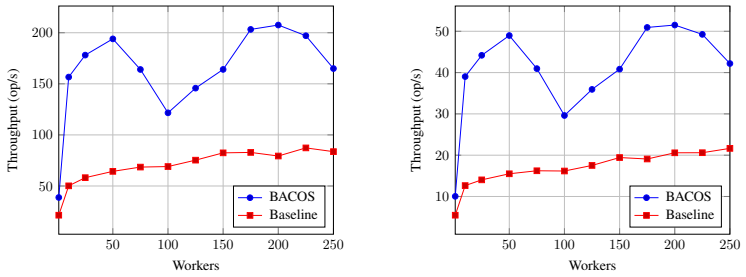
Figure 4. Result of 95% of confidence interval for the mean of difference of response time distributions between baseline and BACOS for read and write operations.

5.2.2. Throughput

To compare the throughput for read and write operations in Figure 5, we computed 95% of confidence interval for the mean of difference of throughput distributions between BACOS and baseline for read and write operations as shown in Figure 6. We can conclude that in both types of operations, our approach was better than the baseline since the zero value was not included in the interval and the sign of the mean was positive, indicating that the BACOS throughput rates are better than baseline. Also, Figures 5 and 6 show

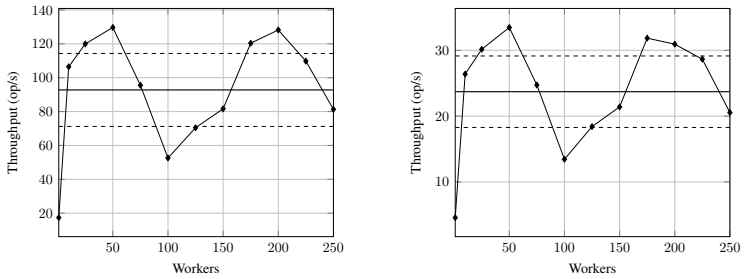
that read and write operations have a similar behavior. As the proportion of number of operation has been set up with 80% to read and 20% to write, it was expected that the results show a throughput much bigger in read operations than in write ones. The best throughput rates of read and write operations were reached by BACOS when the storage handled 200 workers, however the system could not keep providing similar values since it had to deal with a high demand and lack of storage device resources. On the other hand, with same system resources, the baseline strategy wasn't able to reach similar throughput rates because of its inefficiency to manage storage device performance.

Analyzing read throughput in Figure 5(a) and write throughput in Figure 5(b) side by side, the experiments have shown that our strategy was not able to keep providing high throughput rates when the number of workers was between 50 and 150. This behavior was expected, since our approach made the decision to change weight of performance classes, hence it had to execute migration of a large number of object to ensure further performance. While this process is running, it uses more system resources to inner management than to handle the users requests, therefore the system performance decreases not only in number of operations but also in response time as shown in Figure 3. Although the throughput behavior of our strategy was not so smooth compared with the baseline, the experiments still have shown good results regarding to difference of throughput rates for all number of workers as depicted in Figures 6(a) and 6(b).



(a) Read throughput. (b) Write throughput.

Figure 5. Throughput distributions.



(a) Difference of read throughput between BACOS an baseline.

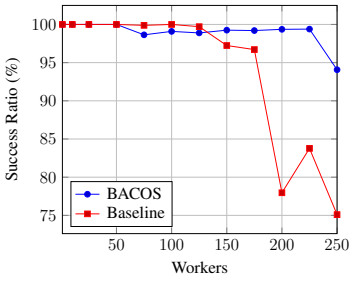
(b) Difference of write throughput between BACOS an baseline.

Figure 6. Result of 95% of confidence interval for the mean of difference of throughput distributions between BACOS and baseline for read and write operations.

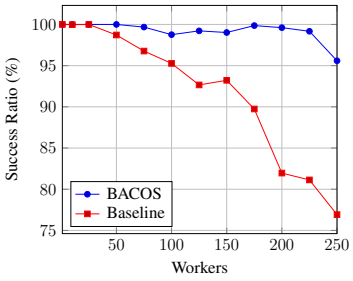
5.2.3. Success Ratio

To compare success ratio in Figure 7 with respect to read and write operations, our strategy was much better than the baseline strategy because in the worst case (workload configured with 250 workers) it kept the success ratio above 94% against 74% from the baseline. From analysis only of the success ratio of read operations in Figure 7(a), when the number of workers was between 50 and 125, we see that the baseline strategy was better than BACOS since it kept 100% of success ratio, although our strategy got very close to reach 100% of success ratio while objects were migrating between storage devices. Once the number of workers rises, the baseline strategy was not able to keep the same success ratio. Even on high demand, BACOS still provide a reasonable success ratio, but the success ratio of baseline decreases drastically since the system could not conclude read operations for more than 175 workers.

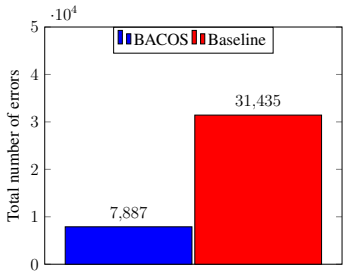
A similar behavior happens to the write operations in Figure 7(b), although the success ratio decreases earlier compared to read operations. The precocious decline of success ratio of write operations in the baseline is justified by the fact that the baseline algorithm considers only capacity, so the strategy will decide to write in storage devices with high available space but not necessarily with best performance. Thus, writing replicas delays new write operations that will be waiting long time in request list of object storage system.



(a) Read Success Ratio.



(b) Write Success Ratio.



(c) Error statistic.

Figure 7. Success ratio comparison.

Note that Figure 7(a) and Figure 7(b) show that as the number of workers increases, it becomes more difficult to maintain the success ratio due to the lack of resources or the bad resource utilization for both strategies. Figure 7(c) show the total number of errors and complements the results of success ratio for read and write operations. The number of errors represents failed operations and express unpredictable and undesirable behaviors. The total number of errors for BACOS was much smaller than baseline strat-

egy once our strategy takes it into consideration.

6. Conclusion and Future Work

This paper presented BACOS, an approach to load balancing in cloud data storage systems that uses previous knowledge of workload behaviour and improves the system performance taking advantage of heterogeneous storage devices. BACOS designates a reasonable amount of requests to faster storage devices without overloading its performance resource. Also, BACOS is a non-intrusive approach that consumes high level interfaces from the storage system to change storage device demand according to a dynamic variation in workload.

In order to evaluate BACOS, experiments that measured response time, throughput and success ratio were conducted in Openstack Swift, a popular commercial object storage. BACOS improved response time and throughput to read/write operations compared to the strategy recommended in official Swift documentation. Additionally, BACOS could provide high success rates even in high demands. Although the experiment evaluated our strategy in a small cluster, our strategy is not limited by the number of the storage nodes. Results corroborate that BACOS improves performance to object storage systems in cloud environments.

There is a number of research opportunities that derive from this work, including: execute experiments with large objects and different ratio of read and write; add support for workload prediction; and incorporate forecast models. Additionally, BACOS can be combined with our previous work about replica selection [Almeida et al. 2016] to improve storage overall system performance. Finally, we intend to conduct a study changing of the number of replicas and consistency issues.

Acknowledgements

This research was partially supported by Hitachi Data Systems (HDS), Funcap/Brazil and LSBD/UFC.

References

Almeida, A. M. R., Cavalcante, D. M., Sousa, F. R. C., and Machado, J. C. (2016). LB-RLT approach for load balancing heterogeneous storage nodes. In *SBRC*.

Chung, H. Y., Chang, C. W., Hsiao, H. C., and Chao, Y. C. (2012). The load rebalancing problem in distributed file systems. In *IEEE Int. Conf. on Cluster Computing*, pages 117–125.

Deshmukh, S. C. and Deshmukh, S. S. (2015). A survey: Load balancing for distributed file system. *International Journal of Computer Applications*, 111(5).

Felber, P., Kropf, P., Schiller, E., and Serbu, S. (2014). Survey on load balancing in peer-to-peer distributed hash tables. *IEEE Communications Surveys & Tutorials*, 16(1):473 – 492.

GlusterFS (2017). Glusterfs. `https://www.gluster.org`. Accessed: 2017-04-03.

Gunawi, H. S., Agrawal, N., Arpaci-Dusseau, A. C., Arpaci-Dusseau, R. H., and Schindler, J. (2005). Deconstructing commodity storage clusters. *SIGARCH Comput. Archit. News*, 33(2):60–71.

Hsiao, H.-C., Chung, H.-Y., Shen, H., and Chao, Y.-C. (2013). Load rebalancing for distributed file systems in clouds. *IEEE Transactions on Parallel and Distributed Systems*, 24(5):951–962.

Jain, R. (1991). *The art of computer systems performance analysis : techniques for experimental design, measurement, simulation, and modeling*. J. Wiley & sons, New York.

Lehmann, T. M., Gonner, C., and Spitzer, K. (1999). Survey: interpolation methods in medical image processing. *IEEE Transactions on Medical Imaging*, 18(11):1049–1075.

Li, J. and Li, B. (2013). Erasure coding for cloud storage systems: A survey. *Tsinghua Science and Technology*, 18(3):259–272.

Mesnier, M., Ganger, G. R., and Riedel, E. (2003). Object-based storage. *IEEE Communications Magazine*, 41(8):84–90.

Nuaimi, K. A., Mohamed, N., Nuaimi, M. A., and Al-Jaroodi, J. (2013). A partial replication load balancing algorithm for distributed data as a service (daas). In *Int. Conf. on High Performance Computing and Simulation (HPCS)*, pages 35–40.

OpenStack (2017). Openstack swift. `http://docs.openstack.org/developer/swift/`. Accessed: 2017-04-03.

Rackspace (2017). Rackspace. `https://www.rackspace.com/`. Accessed: 2017-04-03.

SwiftStack (2017). Swiftstack. `https://www.swiftstack.com/`. Accessed: 2017-04-03.

Tan, Z., Zhou, W., Feng, D., and Zhang, W. (2013). Aldm: Adaptive loading data migration in distributed file systems. *IEEE Transactions on Magnetics*, 49(6):2645–2652.

Wang, Z., Chen, H., Fu, Y., Liu, D., and Ban, Y. (2015). Workload balancing and adaptive resource management for the swift storage system on cloud. *Future Gener. Comput. Syst.*, 51(C):120–131.

Weil, S. A., Brandt, S. A., Miller, E. L., Long, D. D., and Maltzahn, C. (2006). Ceph: A scalable, high-performance distributed file system. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 307–320. USENIX Association.

Zheng, Q., Chen, H., Wang, Y., Zhang, J., and Duan, J. (2013). COSBench: cloud object storage benchmark. In *Proceedings of the 4th ACM/SPEC Int. Conf. on Performance Engineering*, pages 199–210. ACM.

Alocação de Ambientes Virtuais com base na Afinidade entre Perfis de Aplicações Massivamente Paralelas e Distribuídas

Victor Oliveira[1 2], **Jonathan Barbosa**[2], **Matheus Bandini**[2], **Bruno Schulze**[2], **Raquel Pinto**[1]

[1]Instituto Militar de Engenharia (IME)
Rio de Janeiro – RJ – Brasil

[2]Laboratório Nacional de Computação Científica (LNCC)
Petrópolis – RJ – Brasil

{victord,jonathanpb,mbandini,schulze}@lncc.br, raquel@ime.eb.br

Abstract. *This paper presents a virtual machine scheduling algorithm to run Massively Parallel and Distributed Computing applications with intensive usage of CPU, memory and I/O. The scheduling algorithm is proposed in order to address the allocation of virtual machines in Cloud Computing environments based on the concept of "Affinity" between applications. For the method's implementation, the virtual machines resource consumption was monitored, in order to obtain historical data that allows to determine application profiles. As a result, a virtual machine scheduler for physical shared resources was created. Its objectives are to avoid combinations of applications that may cause performance degradation and to improve the usage rate of computing resources.*

Resumo. *Este artigo apresenta um algoritmo de escalonamento de máquinas virtuais que executam aplicações de Computação Massivamente Paralela e Distribuída (CMPD), com uso intensivo de CPU, memória e I/O. O método de escalonamento tem por finalidade tratar a alocação das máquinas virtuais em ambientes de nuvens computacionais com base no conceito de "Afinidade" entre aplicações. Para a implementação do método, foi realizado o monitoramento do consumo de recursos das máquinas virtuais para obter dados históricos de execução que permitem determinar os perfis das aplicações. Como resultado, foi criado um escalonador para alocar máquinas virtuais, cujos objetivos são evitar combinações de aplicações que causem degradação do desempenho entre si e melhorar a taxa de utilização dos recursos computacionais.*

1. Introdução

Na tentativa de melhor aproveitar o uso dos recursos computacionais de modo a reduzir custos, novas técnicas, tecnologias e arquiteturas desenvolvidas estão conquistando grande aceitação no mercado e no meio acadêmico. Uma dessas tecnologias é a Computação em Nuvem, que tenta resolver problemas como consumo energético e alocação de espaço físico em grandes centros de processamento de dados e de CMPD (Computação Massivamente Paralela e Distribuída).

A preocupação crescente com a qualidade dos serviços prestados por provedores de nuvens computacionais motiva pesquisas focadas em desenvolver mecanismos e

metodologias para promover melhorias na forma de alocar aplicações nesses recursos [Zheng et al. 2013]. Neste sentido, conhecer o perfil do consumo de recursos das aplicações, os ambientes virtualizados e os efeitos causados pela concorrência contribui para esses esforços, no sentido de minimizar as perdas de desempenho.

Com o objetivo de otimizar o uso da infraestrutura disponível, um dos fatores que caracterizam a Computação em Nuvem é a possibilidade de haver a competição por um mesmo recurso físico, devido a dois ou mais ambientes virtualizados compartilharem tais recursos. A concorrência, entretanto, pode resultar na degradação das aplicações de nuvem. De acordo com [Emani and O'Boyle 2015], a alocação inadequada de aplicações concorrentes pode causar a degradação do desempenho. Caso os limites especificados por contratos de Qualidade de Serviço sejam extrapolados, a proposta de nuvem pode ser invalidada. Por essa razão, torna-se necessário desenvolver métodos de escalonamento de aplicações de nuvem que permitam alocar aplicações que possuam características diferentes e, por conseguinte, reduzem o impacto da concorrência entre si. Esse princípio é conhecido na literatura como Afinidade entre aplicações [Licht 2014].

Dado que as aplicações em uma máquina virtual podem alterar o seu perfil de uso de recursos computacionais ao longo da execução, é necessário analisar essa mudança no caso em que for gerada degradação em outros ambientes virtuais. Por exemplo, apenas classificar a aplicação como intensiva em processamento (CPU-*Bound*), não permite afirmar que esta máquina virtual vai ocupar 100% de CPU durante todo o tempo de execução. Existe a possibilidade de que, em um dado momento, a aplicação troque seu perfil de consumo e comece a utilizar outro recurso intensamente [Schad et al. 2010]. É neste momento que se apresenta uma das motivações deste trabalho, onde a alteração no perfil da aplicação pode acarretar sobrecarga e degradação na execução das outras máquinas virtuais que estão alocadas em um mesmo hospedeiro.

O uso do conceito de Afinidade entre aplicações [Mury et al. 2014] visa contribuir na alocação do ambiente virtual que se baseia nas características de consumo e afinidade entre aplicações. Sendo assim, é necessário monitorar e analisar os diversos perfis das aplicações, estabelecidos através do histórico do consumo de recursos. Por meio deste estudo, foi definido um grau de afinidade que é utilizado pelo escalonador para otimizar o processo de alocação e migração dos ambientes virtuais em uma nuvem, afim de evitar o impactado da concorrência dos recursos computacionais.

Este trabalho propõe duas técnicas de escalonamento, o estático e o dinâmico. Estas duas técnicas referem-se ao momento em que as decisões são tomadas. No escalonamento estático, os perfis das aplicações são previamente conhecidos e, uma vez escalonadas, as máquinas virtuais são mantidas na mesma máquina física até o fim da execução. Entretanto, no escalonamento dinâmico, pode-se não ter conhecimento inicial sobre as características da aplicação, de forma que o perfil de uso dos recursos pode mudar ao longo da execução. As aplicações chegam ao escalonador em momentos distintos de tempo. Quando o escalonador detecta alteração no perfil da aplicação, ele pode decidir migrar as máquinas virtuais de forma a evitar a queda do desempenho daquelas que compartilham o mesmo ambiente físico [Alam and Varshney 2016].

A seção 2 deste artigo discute alguns trabalhos relacionados aos principais tópicos abordados durante o desenvolvimento do método de escalonamento proposto. A seção 3

apresenta a metodologia utilizada para determinar a Afinidade entre um conjunto de aplicações e entre recursos computacionais. A seção 4 apresenta o método de escalonamento proposto e descreve a arquitetura do escalonador desenvolvido. A seção 5 apresenta os resultados dos experimentos que buscaram validar o algoritmo de escalonamento criado. Por fim, a seção 6 conclui o artigo e apresenta propostas de trabalhos futuros.

2. Revisão da Literatura

Em [Mury et al. 2014], é avaliado o aumento do uso de ambientes virtualizados. Entretanto, a maioria desses estudos são limitados a um nível de análise de desempenho, não aprofundando o estudo sobre os efeitos da concorrência entre os vários ambientes virtuais, nem como mitigar esses efeitos. Esse trabalho apresentou o conceito de Afinidade, que define o grau de coexistência entre as classes de aplicações. As classes de algoritmos associadas aos tipos de bibliotecas paralelas utilizadas na implementação dessas aplicações influenciam nas suas combinações. Os resultados demonstraram que os efeitos dessas combinações têm valores diversos, o que torna necessário detalhar o estudo para classificá-los. Sendo assim, justifica-se definir e analisar o conceito de "Afinidade", buscando melhorar o uso dos recursos, sobretudo no que tange a Computação Massivamente Paralela e Distribuída. Por fim, a principal contribuição foi a criação de tabelas comparativas de desempenho concorrente que permitem a análise visual dos resultados. Dessa forma, é possível avaliar os impactos causados pela concorrência, o que possibilita a rápida avaliação das melhores combinações, bem como daquelas que devem ser evitadas.

O trabalho de [Yokoyama 2015] apresentou uma plataforma de nuvem privada voltada à criação e gerência de *clusters* computacionais para a aplicação na resolução de tarefas de computação de alto desempenho. Foi adotada como base uma arquitetura paralela de memória distribuída. Além de desenvolver um sistema para criação de *clusters* computacionais em nuvem, o trabalho ainda apresentou um modelo de escalonamento *offline* de máquinas virtuais baseado na Afinidade entre as aplicações de *Benchmark* em execução nos hospedeiros. Tal modelo de alocação busca melhorar o aproveitamento dos recursos disponíveis na infraestrutura e a vazão de tarefas executadas.

A abordagem deste trabalho difere da apresentada por [Yokoyama 2015] em três aspectos principais: **i)** o *ProSched* realiza o escalonamento *online* e *offline* das aplicações virtuais; **ii)** o escalonamento proposto foi aplicado tanto para aplicações reais como para *Benchmarks*; e **iii)** possui a capacidade de migrar as aplicações de acordo com a afinidade entre elas, o que não é contemplado por [Yokoyama 2015].

[Bernado 2014] apresenta uma arquitetura capaz de suportar os efeitos de sobrecargas momentâneas em servidores físicos e virtuais hospedados em ambientes de nuvem. A arquitetura, denominada *Phoenix*, tem o objetivo de automatizar a gerência de máquinas virtuais hospedadas em uma nuvem. Além disso, é proposto um esquema de balanceamento de carga nos momentos em que não há sobrecarga dos recursos, associada a uma estratégia reativa, que é acionada caso tais sobrecargas sejam identificadas.

[Calheiros et al. 2011], por sua vez, apresenta o *CloudSim*, cujo objetivo é fornecer um sistema de simulação que permite a modelagem, a simulação e a experimentação de infraestruturas de nuvem e serviços de aplicativos. Dentre as conclusões obtidas, está a identificação da necessidade de monitorar as aplicações, com a finalidade de otimizar o uso da nuvem computacional e verificar os efeitos da concorrência na infraestrutura.

Entretanto não é apresentado um estudo que defina quais tipos de aplicações poderiam coexistir nestes ambientes virtuais, sem que haja a degradação em função da concorrência por recursos computacionais.

No trabalho de [Simmons et al. 2007], é proposta a criação de um sistema de tomada de decisões baseado em SLAs para a agregação de recursos de forma otimizada. Na proposta, há um controle de uso de recursos que pontua os gastos e compara com os níveis de serviços propostos, penalizando a carga excessiva. A proposta dos autores baseia-se no cálculo de consumo, mas refere-se à plataforma como um serviço, sem monitorar a carga em toda a infraestrutura, nem avaliam os diferentes tipos de aplicações que podem concorrer por um mesmo recurso.

[Nanos et al. 2010] apresenta uma análise sobre o impacto das aplicações científicas que são executadas em um *cluster* virtualizado, baseando-se no impacto causado pelo uso intensivo de rede e de I/O. Os resultados e as conclusões dos autores apontam a necessidade de definir o perfil do comportamento das aplicações para melhor escaloná-las em ambientes de HPC virtualizados, afim de evitar a sobrecarga dos recursos computacionais.

O trabalho desenvolvido por [Juliani 2014] apresenta um método de escalonamento de máquinas virtuais em nuvens computacionais focadas em HPC. O escalonador desenvolvido pelo autor leva em consideração o consumo energético e o tipo de carga de trabalho que as máquinas virtuais irão executar para decidir quando e em qual servidor as mesmas serão alocadas. A avaliação do algoritmo foi feita através da utilização do *CloudSim*. Os resultados obtidos indicam a necessidade de analisar detalhes específicos das infraestruturas e aplicações para contribuir na otimização dos recursos e, consequentemente, aumentar os níveis de oferta de serviço e reduzir os problemas causados pelo uso dos recursos de forma concorrente.

Todos os trabalhos anteriormente apresentados apontam para a lacuna ainda existente quanto à necessidade de aprofundar os estudos dos efeitos da concorrência pelo compartilhamento de um ambiente real por vários ambientes virtualizados lá hospedados. Mesmo assim, eles não citam a necessidade de fazer este estudo com a utilização do conceito de Afinidade entre as aplicações.

3. Avaliação Experimental da Afinidade entre Aplicações

As avaliações sobre os efeitos das concorrência são importantes para definir o modelo de escalonamento para uso em ambientes distribuídos. A análise do comportamento das aplicações reais e sintéticas em máquinas virtuais, executando em um ambiente controlado e com equipamentos homogêneos, permitiu verificar o efeito da concorrência e sobretudo a importância do conceito de afinidade. Isso possibilita estender esse modelo para ambientes reais de nuvens. A afinidade é caracterizada por um grau normalizado que define o nível de influência entre aplicações que compartilham recursos. E o perfil é uma análise dinâmica do comportamento de consumo de recursos das aplicações durante o seu ciclo de vida, tais como suas intensividades e seu histórico de consumo.

Foram utilizadas *benchmarks* e aplicações reais científicas, afim de simular um ambiente real e assim avaliar os diversos perfis de aplicações. No decorrer das execuções das aplicações, um histórico do perfil de uso de recursos foi obtido por meio do monitoramento. São utilizados como parâmetros para esta estratégia: uso de CPU, memória, I/0

de disco. Estas informações são utilizadas para obter o perfil das aplicações científicas em ambientes dedicados e compartilhados. Vale frisar que classes de aplicações e perfil são conceitos distintos. Uma classe se difere de perfil uma vez que aplicações de uma mesma classe podem possuir perfis/comportamentos diferentes.

As aplicações utilizadas neste trabalho foram escolhidas por apresentarem perfis distintos de uso de recursos computacionais: HPL (*High-Performance Linpack Benchmark*) é uma aplicação sintética intensiva em CPU e que pode ser intensiva em memória, dependendo do tamanho da matriz de entrada; IOzone é uma aplicação sintética intensiva em I/O de disco que realiza operações sobre um sistema de arquivos; BLAST (*Basic Local Alignment Search Tool*) é uma aplicação real utilizada na área da Biologia e que apresenta uso intensivo de memória; e Montage (*Image Mosaic Software for Astronomers*) é uma aplicação científica utilizada na área da Astronomia cujo perfil de uso de recursos apresenta variações na intensidade de uso ao longo do tempo, o que valida a hipótese de que as aplicações podem sofrer alterações no decorrer da execução.

Para realização dos experimentos, foi utilizado KVM como camada de virtualização, e as seguintes configurações de infraestrutura: 3 servidores reais com Processador Intel(R) CPU X5650 2.67 GHz (12 núcleos), 16 GB de memória RAM, HD de 1TB (7200RPM), Sistema Operacional *Ubuntu Server* 14.04 LTS, rede Gigabit Ethernet. Em cada servidor real são alocados no máximo 2 ambientes virtuais. Isso tem por finalidade avaliar a afinidade entre duas máquinas virtuais concorrendo por recursos reais. Essas máquinas virtuais foram configurados da seguinte maneira: 4 núcleos QEMU Virtual, 6GB de memória RAM, HD Virtual de 20GB e Sistema Operacional *Ubuntu Server* 14.04 LTS. E os experimentos foram divididos em 2 grupos: experimentos executados em ambientes dedicado e ambientes compartilhados. Para cada experimento, foram realizados 30 execuções.

Nos experimentos em ambientes dedicados, todas as aplicações propostas foram executadas de modo isolado, sem qualquer outra aplicação sendo executada na mesma máquina física.

De acordo com [Patterson and Hennessy 2016], "Um computador que processa uma mesma quantidade de carga de trabalho em menos tempo é o mais rápido", definindo assim, tempo de execução como a melhor medida para definir o desempenho computacional. Diante de tal assertiva, o tempo é a medida escolhida para definir o grau da afinidade entre as aplicações executadas em ambientes virtuais, calculado pela Equação 1.

$$A_{i,j} = \frac{Tb_i}{Tc_{i,j}} \tag{1}$$

Nos experimentos em ambientes compartilhados, foram executadas todas as combinações de pares de aplicações, gerando a matriz de afinidades apresentada na tabela 1. Essa matriz é utilizada no escalonamento estático, que utiliza o conhecimento prévio para melhor alocar as aplicações a partir de uma fila de execução. A equação apresentada a cima é usada para obter o grau de afinidade entre as aplicações, a partir dos tempos de execução. A afinidade *A* entre as aplicações *i* e *j* é definida pela razão entre o tempo base (*Tb*) da aplicação *i* e o seu tempo de execução em concorrência (*Tc*) com a aplicação *j*.

Tabela 1. Matriz de afinidade entre as aplicações (quanto maior melhor)

	HPL	**BLAST**	**IOZONE**	**MONTAGE**
HPL	0,83	0,90	0,91	0,95
BLAST	0,85	0,91	0,86	0,91
IOZONE	0,93	0,96	0,51	0,83
MONTAGE	0,85	0,92	0,43	0,57

As aplicações utilizadas neste trabalho foram escolhidas por cada uma usar de forma intensiva um recurso computacional específico. Isso permitiu a simplificação da matriz de afinidade das aplicações para a criação de um nova matriz, mais genérica, baseando-se nos recursos computacionais CPU, Memória e I/O de disco, como pode ser visto na tabela 2. Essa matriz genérica é utilizada no escalonamento dinâmico, onde o perfil de uma aplicação pode não ser conhecido, fazendo com que o escalonador tenha que tomar decisões em tempo de execução.

Tabela 2. Matriz de afinidade baseado nos recursos computacionais

	CPU	**Memória**	**I/O**
CPU	0,83	0,90	0,91
Memória	0,85	0,91	0,86
I/O	0,93	0,96	0,51

Os valores obtidos estão normalizados de forma que $0 \leq a_{i,j} \leq 1$, tal que *a* representa um valor da matriz de afinidade de *i* com *j*. Quanto mais próximo *a* é de 1, melhor é o grau da afinidade. Em contrapartida, quanto menor e mais próximo do zero, pior é a afinidade de *i* com *j*.

4. Escalonador ProSched

O método de escalonamento desenvolvido é constituído por quatro serviços independentes. Cada serviço é responsável por desempenhar uma função específica no método de escalonamento (Figura 1). É importante ressaltar que, no contexto deste trabalho, uma tarefa é caracterizada por uma máquina virtual que executa uma ou mais aplicações.

O serviço `ProSched Web` é a interface através da qual a submissão de aplicações, gerenciamento e o acompanhamento, em tempo real, da alocação e utilização de recursos e tarefas são realizados pelos administradores.

O `Deployer` tem o objetivo de enviar as aplicações para a infraestrutura, atuando diretamente no gerenciamento das máquinas virtuais. Ele trabalha diretamente em conjunto com o escalonador, requisitando a máquina física mais apropriada para a execução de cada aplicação. Após essa comunicação, o `Deployer` inicia instâncias virtuais nos recursos reais ou realiza a migração em tempo real de máquinas virtuais cujos graus de afinidade sejam baixos (inferiores a 0.60). Este serviço também é responsável pela ativação dos monitores de cada máquina virtual da infraestrutura. Vale enfatizar que o patamar de afinidade 0.6 foi obtido de modo empírico através do histórico de execução das aplicações sendo possível ajustá-lo.

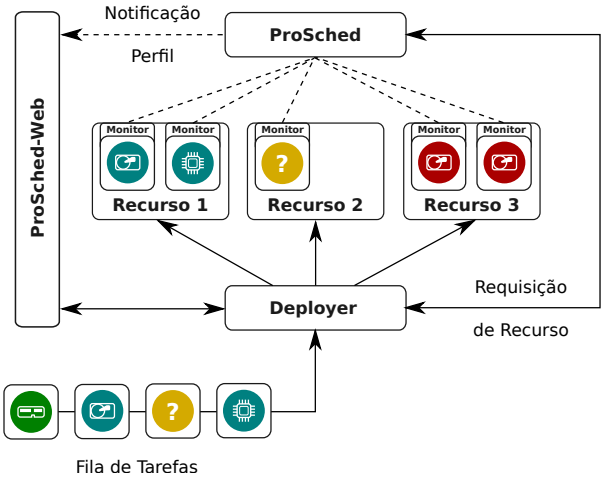


Figura 1. Arquitetura do Escalonador Baseado no Perfil das Aplicações

O serviço `Monitor` tem o objetivo de coletar e analisar dados sobre as tarefas durante a sua execução em uma máquina virtual. Para cada ambiente virtual, um agente monitor é alocado para coletar as informações de uso de recursos e, ao final, armazenar seu histórico em arquivo para ser utilizado posteriormente. Esta coleta é feita de forma não intrusiva, sem a necessidade de modificar o código das aplicações (Figura 2).

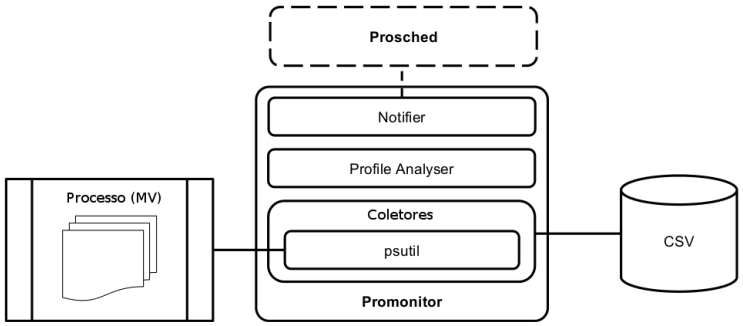
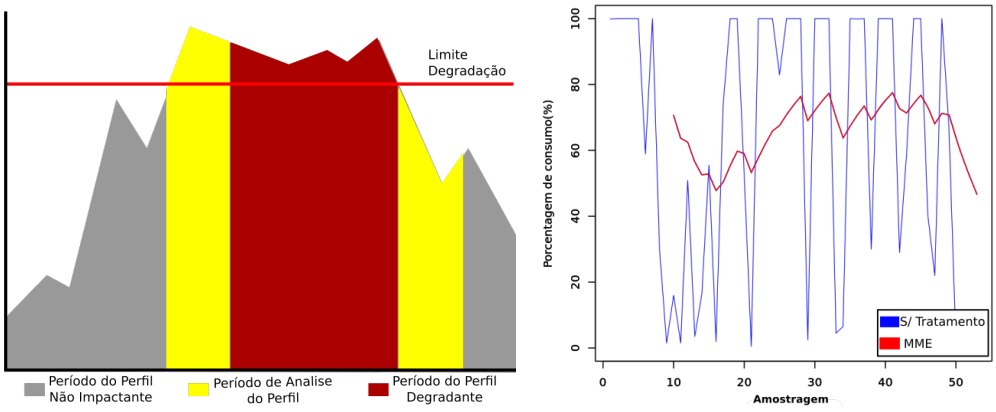


Figura 2. Arquitetura do Serviço Monitor

O perfil de execução das aplicações é obtido pelo monitoramento da tarefa. Uma vez que o valor coletado ultrapasse o limite de degradação do sistema, situação esta identificada pelo `Profile Analyser`, o monitor envia uma mensagem, através do módulo `Notifier`, sinalizando ao escalonador sobre a troca de perfil de consumo de recursos desta tarefa (Figura 3(a)). Durante todo o monitoramento da tarefa, os dados de interesse são coletados e armazenados em arquivos que serão utilizados pelo escalonador como conhecimento base em futuras execuções.

Quando analisado o gráfico de um histórico de consumo de CPU de uma tarefa, é possível verificar imediatamente dois aspectos: a variação do uso dos recursos formam picos e vales, e a existência de tendências ao longo do tempo. Uma solução encontrada para este problema foi a aplicação da Média Móvel Exponencial (MME) nos valores de consumo de recursos obtidos dos ambientes virtuais. Deste modo, os movimentos de curva são suavizados, permitindo uma representação real do comportamento das aplica-

ções. Isso permite que mudanças repentinas, e não constantes, não sejam erroneamente classificadas como o perfil atual da tarefa (Figura 3(b)).



(a) Processo de análise do perfil (b) Coleta com MME e sem tratamento

Figura 3. Método de análise de dados do Monitor

O próximo serviço a ser tratado é o `Prosched`. Sua principal contribuição é reduzir o *makespan*[1] de uma fila de tarefas. Tal contribuição é alcançada por meio do aprendizado de seus perfis dinâmicos com base em execuções anteriores. Em casos de mudança de perfil, o escalonador é capaz de alocar ou migrar a tarefa para outra máquina real na infraestrutura. Para isso, é feita uma análise para encontrar tarefas mais afins, garantindo a manutenção da capacidade de execução desse ambiente.

O comportamento do método de escalonamento desenvolvido une técnicas do algorítimo *Round-Robin* (RR), com afinidade de aplicações e perfil dinâmico de execução. Desta forma, a primeira etapa na alocação é encontrar o recurso com menor quantidade de tarefas. Caso seja encontrado, a tarefa é alocada no recurso disponível. Se os recursos possuírem ao menos uma tarefa, utiliza-se a afinidade de aplicações dada pela tabela 2.

Após uma tarefa ser inserida na fila de execução, o escalonador inicia o processo de eleição de recursos para a sua alocação. Os recursos são organizados de forma a simplificar uma alocação RR, ordenando-os de forma crescente pela quantidade de tarefas. Em paralelo, o escalonador procura em sua tabela de afinidade tarefas iguais e as agrega de forma a obter o perfil médio de execução. Para o conhecimento de execução desse perfil, o escalonador utiliza apenas execuções nas quais a tarefa não concorreu por recursos com outras aplicações. Para isso, os seguintes casos são analisados:

1. A tarefa possui afinidade e não está em execução: o recurso é alocado e o escalonador informa ao `Deployer` em qual *host* iniciar a tarefa. O `Deployer`, por sua vez, inicia a máquina virtual e o monitor para aquela tarefa;
2. A tarefa possui afinidade e está em execução: neste caso, o escalonador apenas registra o perfil de execução da tarefa, sem atuar sobre o sistema;
3. A tarefa não possui afinidade, mas está em execução: o escalonador avalia a tarefa e, com base no perfil médio de execução e se o tempo de execução seja maior do que o tempo de migração, o escalonador requisita ao `Deployer` que realize

[1]Intervalo de tempo entre a alocação da primeira tarefa até o fim da execução da última [Pinedo 2008]

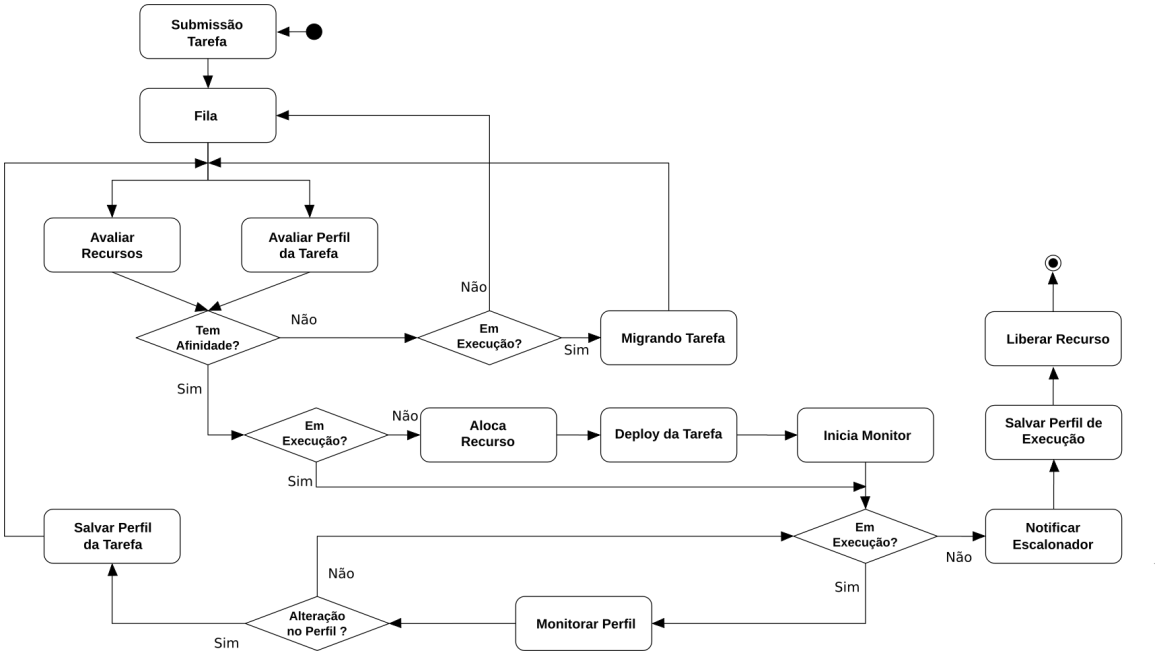


Figura 4. Fluxograma do Algoritmo de Escalonamento Prosched

a *Live-Migration* da máquina virtual para um recurso que a tarefa possua maior afinidade. Caso contrário, não é feita a migração;

4. A tarefa não possui afinidade e não está em execução: O escalonador a coloca na fila para que seja reavaliada durante o processo de notificação dos monitores.

Durante o ciclo de vida da tarefa, o `Monitor` coleta as informações e notifica o escalonador caso uma mudança de perfil seja detectada. Quando a tarefa termina sua execução, o escalonador armazena o perfil na tabela de conhecimento e encerra sua execução, informando que o recurso foi liberado. Na Figura 4 demonstra as etapas do funcionamento do algoritmo de escalonamento desenvolvido.

5. Validação e Resultados

Esta seção tem por objetivo apresentar os resultados do algoritmo de escalonamento estático e dinâmico, bem como validar a estratégia de alocação baseada na afinidade entre as aplicações e entre os recursos computacionais. Para avaliar o desempenho do escalonamento estático, este foi comparado com as estratégias FCFS (*First-Come First-Served*), o *Random*, na forma de uma alocação aleatória, e a estratégia proposta por este trabalho, denominada *Affinity*. Para validação do algoritmo dinâmico, será aplicada uma comparação entre os algoritmo *Round-Robin Affinity*, com e sem conhecimento dos perfis das aplicações, e aplicando sobre uma fila, um conhecimento híbrido, mesclando essas duas possibilidades quanto ao conhecimento dos perfis. As abordagens estática e dinâmica foram adotadas em função dos tipos de escalonamento online e offline.

Com o intuito de validar a hipótese de que a alocação baseada na afinidade pode minimizar o *makespan* e, consequentemente, otimizar o uso dos recursos computacionais, quando comparada com demais abordagens, utilizou-se no experimento estático apenas um servidor real para execução de uma fila de aplicações. Essa configuração teve por finalidade demostrar que, no pior caso, é possível a redução do tempo de uma fila de

execução. O ganho utilizando apenas 1 servidor indica que é possível obter ganhos com mais de um. É importante lembrar que o estudo da afinidade aqui proposto visa avaliar o impacto entre duas máquinas virtuais concorrendo por recursos computacionais em um mesmo hospedeiro. Por isso em ambos os experimentos, são sempre alocadas duas máquinas virtuais, que estarão executando uma ou mais aplicações.

A ordem da fila de execução (Figura 5) é definida pelas seguintes aplicações: BLAST, HPL, IOzone, IOzone, HPL, HPL, Montage, HPL, Montage e IOzone. Esta fila foi criada com o objetivo de analisar os resultados do escalonamento estático envolvendo aplicações com baixo grau de afinidade, de acordo com a Tabela 1.

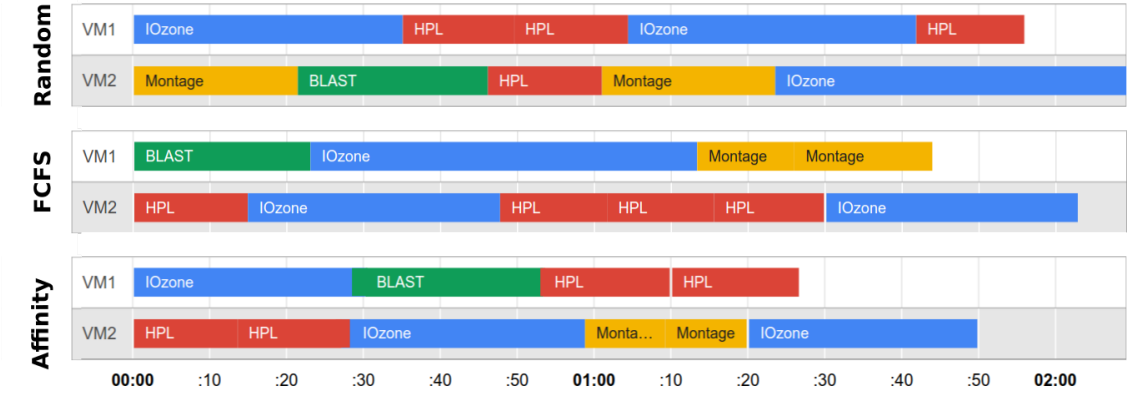


Figura 5. *Timeline* da fila, em horas, da execução de cada estratégia: FCFS, Affinity e Random

A estratégia *Random* foi aquela que resultou no pior desempenho, com um *makespan* médio de 129,5 minutos. O aumento do tempo de execução está relacionado ao fato de que as primeiras aplicações a serem alocadas concorrentemente foram o IOzone e o Montage. A matriz de afinidade mostra um baixo grau de afinidade entre essas aplicações. Com a análise do *timeline* do algoritmo *Random* (Figura 5), é possível verificar que, durante a execução do IOzone, um outro IOzone é escalonado para concorrer pelos mesmo recursos, enquanto que a aplicação seguinte ao IOzone é o HPL. Se o conhecimento do comportamento dessas aplicações tivesse sido utilizado, o HPL poderia ter sido executado antes do IOzone, o que garantiria um melhor uso dos recursos computacionais, alcançando-se também a redução do tempo de execução.

Ao analisar o tempo de cada abordagem (Figura 5), a estratégia baseada na afinidade entre aplicações obteve um *makespan* médio de 110,9 minutos. A redução de tempo está relacionada ao uso do conhecimento prévio para evitar alocações de aplicações com baixo grau de afinidade, o que não ocorreu nos outros algoritmos. Isso foi possível ao evitar que aplicações intensivas em I/O fossem alocadas concorrentemente.

Em resumo, a alocação baseado na afinidade conseguiu reduzir o tempo do *makespan* entre as abordagens FCFS e Random em, aproximadamente, 13,3 e 18,6 minutos, respectivamente. Os resultados constatam que a estratégia proposta obteve um ganho de desempenho de até 16,7%, o que comprova a eficiência do algoritmo.

O experimento cujo resultado é ilustrado pela Figura 6 comprova que o escalonador dinâmico tem a capacidade de evitar a alocação de aplicações com baixo grau de afinidade em situações nas quais os perfis sejam definidos. O experimento visa também

mostrar como o escalonador atua em conjunto com o `Monitor`, quando não há nenhum conhecimento sobre as aplicações, sendo necessário migrá-las através do *live-migration*.

Para os experimentos, foram elaborados perfis de aplicações a serem executados, criados a partir da combinação das aplicações estudas neste trabalho. O objetivo da elaboração desses perfis é validar o emprego, em conjunto, do monitoramento e do escalonamento, visto que as aplicações possuem perfis distintos e, com isso, é mais flexível verificar seu comportamento ao longo das execuções. Isso permite validar a política de escalonamento proposta. Os experimentos foram organizados da seguinte maneira: são enviadas 6 conjuntos de perfis de aplicações para o escalonador, chegando em momentos distintos de tempo, uma após a outra, como acontece em um cenário comercial real. O conjunto de perfis de aplicações elaboradas para o escalonamento são: A {`HPL, IOZONE`}, B {`BLAST, HPL`}, C {`HPL`}, D {`IOZONE`}, E {`HPL, IOZONE`} e F {`BLAST, HPL`};

A fila a ser executada dinamicamente pelo escalonador segue a seguinte ordem de chegada dos Perfis: A, B, D, E, C, F. A formação dessa fila de execução tem por objetivo explorar o pior caso para o algoritmo de escalonamento proposto, por coincidir a alocação concorrente de duas aplicações conflitantes, identificadas nos resultados da análise do efeito da concorrência.

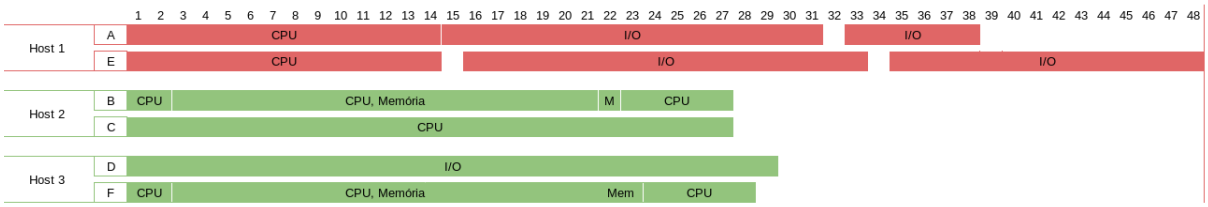


Figura 6. Perfil de execução das tarefas utilizando o método *Round-Robin*(RR)

No primeiro experimento, denominado RR Base, é empregado a estratégia de alocação *Round-Robin*. Essa estratégia de alocação é comumente utilizada por sistemas de nuvens computacionais, como por exemplo, o *OpenStack* [Openstack 2016]. O objetivo deste experimento é comparar essa estratégia com a desenvolvida neste trabalho, que otimiza as alocações posteriores com base no perfil de consumo das aplicações, bem como na migração do ambiente quando o grau de afinidade for baixo.

Através da análise da Figura 6, é possível perceber que o maior impacto foi ocasionado no *Host 1*. Inicialmente, as aplicações eram intensivas em CPU. Entretanto, após aproximadamente 14 minutos, ambas as aplicações trocaram seu perfil de consumo de recursos e passaram a ser intensivas em I/O. Devido a isso, as aplicações acabam competindo pelo mesmo recurso que demonstrou ser o mais crítico em relação à concorrência. O compartilhamento de I/O pelas aplicações causa uma degradação de aproximadamente 50% na execução deste perfil, obtendo assim, um *makespan* de 48 minutos.

A Figura 7 ilustra o experimento no qual são descobertos os perfis das aplicações em tempo real, através do monitoramento e da identificação pelo Monitor. O problema identificado no experimento base (Figura 6) é solucionado pelo escalonador proposto, através da migração dos ambientes virtuais conflitantes. Na Figura 7, é possível perceber que o momento de impacto foi identificado quando o processo de balanceamento de carga é iniciado. Esta abordagem obteve um *makespan* de 41 minutos, 7 minutos a menos do

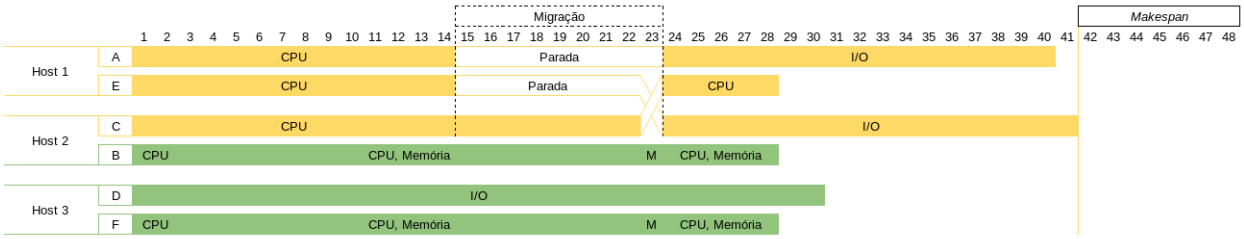


Figura 7. Perfil de execução das tarefas utilizando o método proposto, sem o conhecimento das aplicações (ASC)

que para o RR Base.

O experimento ilustrado pela Figura 8 aloca as aplicações de acordo com os perfis obtidos previamente. Isso permite que a utilização dos recursos seja otimizada nos recursos computacionais, respeitando a matriz de afinidade como base(Tabela 2).

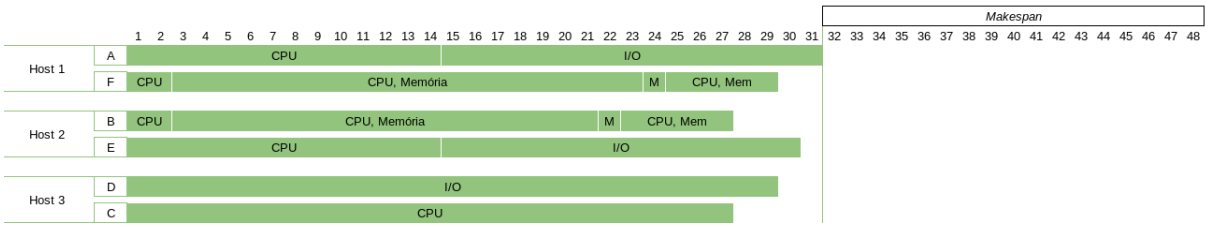


Figura 8. Perfil de execução das tarefas utilizando o método proposto com o conhecimento das aplicações (ACC)

Na figura 8, é possível verificar que, a partir da alocação inicial das 3 primeiras tarefas, inicia-se o uso da matriz de afinidade. É nesse momento que o escalonador verifica o perfil e o histórico de consumo de recursos de cada aplicação. Por exemplo, quando o `Perfil E` é recebido pelo escalonador, verifica-se a existência de uma aplicação com consumo intensivo em I/O. Por esse motivo, os *Hosts* 1 e 3 foram considerados inelegíveis para recebimento de tal tarefa, o que fez com que o escalonador optasse pelo *Host* 2. Em seguida, a próxima aplicação recebida é aquela classificada com o `Perfil C`, que apresenta consumo intensivo de CPU. De acordo com a matriz de afinidade, CPU tem grau de afinidade com I/O de 0,91, razão pela qual o escalonador aloca esta aplicação concorrendo com o `Perfil D` no *Host* 3. As últimas aplicações a serem escalonadas (`Perfil F`) possuem intensividade de CPU e de Memória. Com isso, ela é alocada concorrendo com I/O, devido ao grau de afinidade com aplicações deste tipo ser de 0,86.

O algoritmo proposto permitiu aliar o estudo do impacto da concorrência entre aplicações, em conjunto com o conhecimento sobre os seus perfis. A adoção do algoritmo resultou na redução do *makespan* de aproximadamente 31 minutos, sendo, em média, 54% mais rápido do que o Experimento 1(RR), e 38% quando comparado ao algoritmo com uso da afinidade sem conhecimento das aplicações, apresentado no Experimento 2 (ASC).

Os resultados destes experimentos permitiram demonstrar o ganho de tempo quando utiliza-se perfis de aplicações. Além disso, conforme as aplicações são repetidamente executadas, mais refinado fica o seu perfil, o que permite aumentar a qualidade

do escalonamento em alocações futuras.

6. Conclusões e Trabalhos Futuros

Neste trabalho foi desenvolvido e apresentado um algoritmo de escalonamento de máquinas virtuais, com o objetivo de utilizar o conceito de afinidades entre as aplicações, a fim de aumentar a taxa de uso dos recursos computacionais. Para tanto, foi desenvolvido um sistema de monitoramento de máquinas virtuais, não intrusivo, que analisa e define os diversos perfis das aplicações. O monitor utiliza um analisador de média móvel exponencial que evita a notificação errônea da variação dos perfis da aplicação.

Uma vez que três das aplicações estudadas apresentam uso intensivo de recursos distintos, foi possível criar uma generalização da matriz de afinidades entre aplicações para uma matriz de afinidade entre recursos que, por sua vez, é adotada pela estratégia de escalonamento dinâmico desenvolvida.

Os resultados comprovam os benefícios de utilizar a estratégia de escalonamento *Affinity*, que explora a concorrência entre as aplicações que possuem maior grau de afinidade. Os resultados do uso do modelo associado ao escalonamento estático, proporcionaram um ganho de até 16,7% de desempenho em relação às outras implementações, equivalente à redução de aproximadamente uma hora no *makespan*.

Os resultados obtidos no experimento de alocação dinâmica comprovam a eficiência do escalonador desenvolvido. Foi possível agregar o estudo sobre afinidade das aplicações ao sistema de monitoramento, para identificar uma tarefa que altera o seu consumo de recursos computacionais e impacta negativamente sobre a infraestrutura. Além disso, permitiu a identificação dos perfis em tempo real, além da analise do histórico de consumo, otimizando as alocações e consequentemente, o uso dos recursos computacionais. Um outro ponto que merece destaque é o sistema que identifica e migra os ambientes virtuais quando uma aplicação com baixo grau de afinidade é identificada. Esse fator, aliado à alocação eficiente, permitiram ao escalonador proposto obter melhor uso dos recursos, além de reduzir o tempo que uma aplicação espera na fila até ser executada. Foi possível constatar uma redução do *makespan* de, respectivamente, 17% (7 minutos) do ASC e de 54,8% (17 minutos) do ACC, respectivamente, em relação ao escalonamento RR Base.

Com o desenvolvimento apresentado neste artigo, foram identificados tópicos que podem ser explorados na forma de trabalhos futuros. São eles: i) aperfeiçoar o método de escalonamento para que sua política de alocação trabalhe com atribuição de prioridades, de modo que uma aplicação crítica seja executada com antecedência; ii) avaliar o grau de afinidade entre três ou mais máquinas virtuais em um mesmo servidor físico; iii) analisar o perfil das aplicações e o impacto sobre a afinidade entre aplicações em ambientes com arquiteturas heterogêneas; iv) avaliar o impacto da concorrência por recursos de rede sobre o modelo de escalonamento proposto, em especial, no que se refere à migração das máquinas virtuais.

Referências

Alam, M. and Varshney, A. K. (2016). A New Approach of Dynamic Load Balancing Scheduling Algorithm for Homogeneous Multiprocessor System. *International Journal of Applied Evolutionary Computation (IJAEC)*, 7(2):61–75.

Bernado, E., P. W. P. R. (2014). Arquitetura para Suportar Sobrecargas Momentâneas em Ambientes de Computação em Nuvem. (dissertação de mestrado), Instituto Militar de Engenharia (IME).

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. (2011). CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software: Practice and Experience*, 41(1):23–50.

Emani, M. K. and O'Boyle, M. (2015). Celebrating Diversity: A Mixture of Experts Approach for Runtime Mapping in Dynamic Environments. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2015, pages 499–508, New York, NY, USA. ACM.

Juliani, F. (2014). Um Método de Escalonamento Baseado no Comportamento de Aplicações HPC para Nuvens Computacionais Balanceando Desempenho e Eficiência Energética. (dissertação de mestrado), Instituto Militar de Engenharia.

Licht, F. L. (2014). *Afinidade de Tipos de Aplicações em Nuvens Computacionais*. PhD thesis, Universidade Federal do Parana, Departamento de Informatica, Curitiba, PR.

Mury, A. R., Schulze, B., Licht, F. L., de Bona, L. C., and Ferro, M. (2014). A Concurrency Mitigation Proposal for Sharing Environments: An Affinity Approach Based on Applications Classes. In *Intelligent Cloud Computing*, pages 26–45. Springer.

Nanos, A., Goumas, G., and Koziris, N. (2010). Exploring I/O Virtualization Data Paths for MPI Applications in a Cluster of VMs: a Networking Perspective. In *European Conference on Parallel Processing*, pages 665–671. Springer.

Openstack (2016). Openstack Documentation Review Associate VM Placement. `http://docs.openstack.org/icehouse/training-guides/content/operator-computer-node.html`. Accessed: 2016-11-28.

Patterson, D. and Hennessy, J. (2016). *Computer Organization and Design: The Hardware Software Interface: ARM Edition*. The Morgan Kaufmann Series in Computer Architecture and Design. Elsevier Science.

Pinedo, M. L. (2008). *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 3rd edition.

Schad, J., Dittrich, J., and Quiané-Ruiz, J.-A. (2010). Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance. *Proceedings of the VLDB Endowment*, 3(1-2):460–471.

Simmons, B., McCloskey, A., and Lutfiyya, H. (2007). Dynamic Provisioning of Resources in Data Centers. In *Autonomic and Autonomous Systems, 2007. ICAS07. Third International Conference on*, pages 40–40. IEEE.

Yokoyama, D. (2015). Modelo para o Escalonamento de Aplicações Científicas em Ambientes de Nuvens Baseado em Afinidades. Dissertação de Mestrado, Laboratório Nacional de Computação Científica.

Zheng, Z., Wu, X., Zhang, Y., Lyu, M. R., and Wang, J. (2013). QoS Ranking Prediction for Cloud Services. *IEEE Trans. Parallel Distrib. Syst.*, 24(6):1213–1222.

Alta Disponibilidade de um Gerenciador de Nuvem IaaS Baseada em Replicação: Experiência & Resultados

Gustavo B. Heimovski[1], Rogério C. Turchetti[1], Juliano A. Wickboldt[2], Lisandro Z. Granville[2], Elias P. Duarte Jr[1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19.018 – 81.531-980 – Curitiba – PR – Brasil

[2]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre - RS - Brasil

{gbheimovski,rcturchetti,elias}@inf.ufpr.br

{jwickboldt,granville}@inf.ufrgs.br

Abstract. *In this work we report the experience of adding a high availability solution to an IaaS (Infrastructure as a Service) cloud platform called Aurora. The proposed solution is based on the multi-master replication of the cloud manager, replicas form a cluster of cloud managers running on multiple datacenters. The implementation also includes a monitoring service for the replicas. The high availability solution is fully integrated to the cloud platform so that activation is done directly by pushing a button of the Graphical User Interface. The performance and robustness of the proposed solution were evaluated experimentally. The time to (1) incorporate a new manager instance to a cluster; (2) recover an instance after a failure and (3) replicate data in different scenarios were measured and are reported in this paper. The impact of the proposed solution cost is evaluated by measuring CPU and network usage. A stress test in which the link delay between two datacenters grows up to the operating limit of the replication solution is also reported. Finally, we present a table for the system availability as a function of the MTBF (Mean Time Between Failures).*

Resumo. *Este trabalho relata a experiência de acrescentar uma solução de alta disponibilidade à plataforma Aurora de gerência de recursos de nuvens IaaS (Infrastructure as a Service). A solução proposta facilita a ativação da alta disponibilidade, disparando a replicação multi-master do gerenciador, criando um cluster de gerenciadores sobre múltiplos datacenters. A solução também provê o monitoramento das múltiplas instâncias replicadas. A ativação de alta disponibilidade é feita diretamente na interface do gerenciador. O desempenho e a robustez da solução proposta foram avaliados experimentalmente. São reportados os tempos medidos para (1) incorporar uma nova instância do gerenciador a um cluster; (2) a recuperação após a ocorrência de uma falha e (3) a latência para a replicação dos dados, em diversas configurações. O custo da solução proposta foi avaliado em termos de uso de processador e rede. É reportado também um teste de stress em que o atraso de comunicação entre dois datacenters cresce até o limite de funcionamento da solução de replicação utilizada. Por fim, é apresentada uma tabela com a avaliação da disponibilidade do sistema como função do MTBF (Mean Time Between Failures).*

1. Introdução

A computação em nuvem permite, de forma conveniente, o acesso sob demanda a um conjunto compartilhado de recursos configuráveis de computação, incluindo, redes, servidores, unidades de armazenamento, aplicações e serviços [Armbrust et al. 2010, Zhang et al. 2010]. Estes recursos podem ser rapidamente provisionados e liberados com um esforço mínimo por parte do cliente. Um dos modelos de serviço da computação em nuvem é a Infraestrutura como Serviço (*Infrastructure as a Service* - IaaS), que provê ao usuário recursos como processamento, armazenamento e rede para execução de suas aplicações e sistemas [Mell and Grance 2011, Jhawar and Piuri 2012, Tsakalozos et al. 2011].

Atualmente, observam-se variados tipos de serviços oferecidos pelos provedores de nuvens computacionais, por exemplo, próximo de 61% das atividades comerciais no Reino Unido utilizam algum tipo de serviço em computação em nuvem [Arean 2013]. Entretanto, ainda existem diversos desafios que precisam ser tratados, como: segurança, gerenciamento de riscos, e a alta disponibilidade [Kholghi et al. 2014]. A alta disponibilidade é essencial para permitir que aplicações críticas façam uso da nuvem. Em particular, o gerenciador de nuvem sempre deve se manter acessível para que o cliente ou administrador da nuvem consiga utilizá-lo a qualquer instante. O cliente deve ser capaz de acessar os dados de qualquer gerenciador a qualquer momento, da forma mais transparente possível.

Desta forma, é desejável que uma infraestrutura em nuvem ofereça soluções que permitam o acesso aos recursos mesmo diante de falhas. A proposta do presente trabalho surge neste contexto: desenvolver uma infraestrutura de nuvem robusta, baseada na replicação do gerenciador, que seja ativada de forma simples, diretamente na interface do usuário. Apesar de existirem plataformas de nuvem que possuem mecanismos para tal replicação, todas exigem profundo conhecimento do sistema e um exaustivo esforço para configurar todos os procedimentos até a sua execução.

A estratégia proposta no trabalho envolve a replicação de diversos componentes que mantêm o gerenciador da nuvem disponível, mesmo diante de possíveis falhas. Desta forma, a estratégia permite a replicação e o monitoramento das múltiplas instâncias do gerenciador replicadas. A solução proposta contempla tanto cenários em que há um único *datacenter* com duas ou mais instâncias replicadas da nuvem, quanto cenários em que há dois ou mais *datacenters*, cada um com recursos próprios. Nos dois cenários, cada instância do gerenciador possui acesso a todos os recursos da nuvem. Em caso de falha afetando uma das instâncias, outra instância do gerenciador tem a capacidade de gerenciar seus recursos da nuvem.

No cenário em que existem dois ou mais *datacenters* remotos (conectados pela Internet), a principal motivação é que, caso ocorra falha em um dos *datacenters*, outras instâncias consigam acessar os recursos do *datacenter* que falhou. A ideia de permitir que um gerenciador tenha completo acesso de todos os recursos, não é somente para o caso em que acontecem falhas, mas para que também seja possível utilizar os recursos do outro *datacenter* a qualquer momento, permitindo o compartilhamento de recursos entre nuvens. Além disso, a estratégia proposta também realiza o balanceamento de carga entre vários gerenciadores dos *datacenters*.

A infraestrutura em nuvem utilizada para implementação da estratégia proposta é

denominada de *Aurora Cloud Manager* [Wickboldt et al. 2014]. O Aurora permite o gerenciamento dos recursos da nuvem de forma flexível pela adição de 'programabilidade'. A programabilidade é obtida através de uma API orientada a objetos, na qual os administradores da nuvem conseguem descrever e executar programas personalizados para a otimização e implantação da própria infraestrutura. Para tornar robusto o gerenciador, é implementada uma solução de replicação que utiliza uma abordagem de replicação *multi-master* [Charron-Bost et al. 2010]. A replicação é realizada em duas partes. A primeira parte é executada através da replicação do banco de dados que armazena informações importantes para realizar as ações de gerenciamento. A segunda parte da replicação é realizada utilizando a sincronização de diretórios. A replicação de diretórios é necessária, pois as imagens das máquinas virtuais, os programas de otimização e implantação, além das métricas, ficam armazenados em diretórios distintos, e não no banco de dados.

O desempenho e a robustez da solução proposta foram avaliados experimentalmente. São reportados os tempos medidos para (1) incorporar uma nova instância do gerenciador a um cluster; (2) a recuperação após a ocorrência de uma falha e (3) a latência para a replicação dos dados, em diversas configurações. O custo da solução proposta foi avaliado em termos de uso de processador e rede. É reportado também um teste de stress em que o atraso de comunicação entre dois datacenters cresce até o limite de funcionamento da solução de replicação utilizada. Por fim, é apresentada uma tabela com a avaliação da disponibilidade do sistema como função do MTBF (*Mean Time Between Failures*).

O restante deste trabalho está organizado da seguinte forma. Na seção 2 são descritos ambientes para execução e gerenciamento de recursos em nuvens computacionais e, com base nisto, é feita uma análise comparativa. Na seção 3 é descrito o gerenciador de nuvem *Aurora Cloud Manager*. Na seção 4 são apresentadas as definições e o modelo do gerenciador Aurora Robusto. Os experimentos realizados e suas análises são apresentados na seção 5. A seção 6 apresenta a conclusão e os trabalhos futuros.

2. Trabalhos Relacionados

Nesta seção são descritos ambientes para execução e gerenciamento de recursos em nuvens computacionais, com foco em suas estratégias de alta disponibilidade.

O OpenStack [1] é uma plataforma IaaS em nuvem que oferece capacidade para controlar um grande número de recursos e sua infraestrutura oferece a utilização dos recursos sob demanda. O OpenStack possui duas estratégias para a sua alta disponibilidade [2]. A primeira utiliza a replicação *master-slave*, utilizando o DRBD[3] para replicação do banco de dados. A segunda estratégia utiliza a replicação *multi-master*, onde o banco de dados é replicado através do Galera. Nesta estratégia também é utilizado um balanceador de carga, como o *HAProxy*.

O CloudStack [4] é uma plataforma que possibilita o desenvolvimento de aplicações e gerenciamento de recursos em uma infraestrutura IaaS. Para aumentar a disponibilidade do gerenciador, o próprio operador da nuvem pode replicar o servidor de gerenciamento.

[1] http://openstack.org/
[2] http://docs.openstack.org/ha-guide/
[3] http://drbd.linbit.com/
[4] https://cloudstack.apache.org/

Entretanto, eventos de datacenters diferentes não são integrados e a replicação deve ser habilitada em cada datacenter [5]. Em relação a disponibilidade do sistema de armazenamento, para evitar um único ponto de falha, pode ser implementado em cada zona, um *primary database* e várias réplicas que são sincronizadas com a réplica primária.

O Eucalyptus [6] é uma infraestrutura IaaS projetada para implementar, gerenciar e manter nuvens tanto privadas quanto híbridas. A arquitetura oferece mecanismos para tolerar falhas no nível dos hosts através da substituição imediata do host inoperante, tornando transparente a falha para o usuário. A alta disponibilidade dos serviços do gerenciador estão mapeadas para futuras versões do gerenciador [7], mas ainda não estão totalmente implementadas.

O OpenNebula [8] é uma plataforma em nuvem que traz diversas funcionalidades através de uma arquitetura simples, mas com diversos recursos para a construção de nuvens híbridas. O OpenNebula fornece um conjunto de ferramentas (*toolkit*) para o gerenciamento de infraestruturas heterogêneas sendo compatível com diversas plataformas e hipervisores. O OpenNebula possui suporte para tolerância a falhas e recuperação dos hosts ou das máquinas virtuais que deixarem de executar suas funções prematuramente. O OpenNebula utiliza somente a abordagem *master-slave* para a sua própria replicação. O banco de dados é replicado com o DRBD.

3. O Gerenciador de Nuvens IaaS Aurora

O *Aurora Cloud Manager* [Wickboldt et al. 2014] é uma plataforma de gerenciamento de nuvem que provê infraestrutura como um serviço e permite o gerenciamento dos recursos da nuvem de forma flexível pela adição de 'programabilidade'. A programabilidade é obtida através de uma API orientada a objetos, na qual os administradores da nuvem conseguem descrever e executar programas personalizados para otimização e implantação da própria infraestrutura.

Os recursos providos por uma plataforma de gerenciamento de nuvem são de três tipos básicos: computação, armazenamento e rede. O gerenciamento de recursos de computação é relacionado com a manipulação das máquinas virtuais, que devem ter CPU e memória alocados, por exemplo. A gerência de armazenamento envolve tarefas relacionadas com a alocação de volumes para armazenamento de dados persistentes e, por último, a gerência de recursos de rede é responsável por permitir a comunicação entre os recursos virtuais.

Para prover os recursos de computação e armazenamento, o Aurora suporta tecnologias de virtualização através da *Libvirt*[9], biblioteca responsável por implementar a comunicação com diversos *hypervisors*. Entre os *hypervisors* suportados estão

[5]http://docs.cloudstack.apache.org/projects/cloudstack-administration/en/4.8/reliability.html#limitations-on-database-high-availability

[6]http://www.eucalyptus.com/

[7]https://docs.eucalyptus.com/eucalyptus/4.0.2/install-guide/ha_planning.html

[8]https://opennebula.org/

[9]http://www.libvirt.org

KVM/QEMU[10], *XEN*[11], *LXC*[12], *VirtualBox*[13] e *VMware*[14]. Para prover os recursos de rede, o Aurora utiliza as redes definidas por software (*Software Defined Networks SDN*) através do *Openflow* [McKeown et al. 2008]. Existe também uma integração com o *framework* de monitoramento dos recursos da nuvem *FlexACMS* [Barbosa de Carvalho et al. 2013] e o tradicional sistema de gerência de redes *Nagios*[15].

Na arquitetura do Aurora existem dois tipos de usuários, o usuário final e o administrador do sistema, como vistos no topo da figura 1, que ilustra a arquitetura do gerenciador de nuvem. Os principais módulos do Aurora incluem, o módulo da interface gráfica, *Graphical User Interface (GUI)*, que é composta pela interface com o usuário e a interface com o administrador. O segundo módulo é o *Slice Space*, que possui o componente responsável por transformar as requisições do usuário em um formato interno. Outro módulo é o *Programmable Space*, que possui os componentes responsáveis pela programabilidade do Aurora. Existem outros módulos que são: o *Event Space*, que gerencia eventos, expressos em forma de condições ou alarmes; a *Unified API*, que é responsável pela manipulação dos recursos virtuais, como máquinas virtuais, interfaces de rede, alocação de volumes virtuais, entre outros; E, por último, os *Drivers*, que fazem a abstração dos dispositivos virtuais.

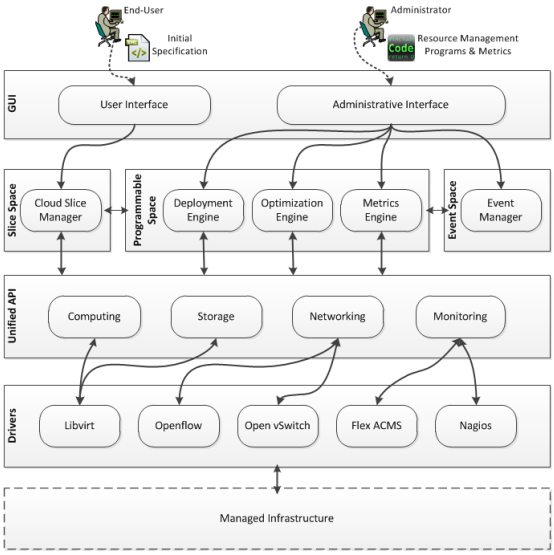


Figura 1. Arquitetura da plataforma de Gerenciamento de nuvem Aurora [Wickboldt et al. 2014].

O componente *Cloud Slice Manager* do módulo *Slice Space* transforma o documento de especificação inicial em um formato interno, chamado *Cloud Slice*. Um *Cloud Slice* agrega todos os diferentes tipos de recursos (computação, armazenamento e rede) que compõem a infraestrutura virtual sobre a qual a aplicação é implantada. Após a criação de um *slice*, os componentes dos módulos *Programmable Space*, *Event Space*

[10]http://www.linux-kvm.org/,http://qemu.org
[11]http://xenserver.org/
[12]https://linuxcontainers.org/
[13]https://www.virtualbox.org/
[14]http://www.vmware.com/
[15]http://www.nagios.org/

e *Slice Space* da arquitetura interagem para organizar o gerenciamento de recursos. Os componentes da arquitetura que adicionam flexibilidade diretamente ao núcleo da plataforma são *Deployment Engine*, *Optimization Engine* e *Metrics Engine*, presentes no módulo *Programmable Space*. Estes componentes operam com base nos programas e métricas de gerenciamento de recursos, na figura: *Resource Management Programs & Metrics*, escritos pelo administrador.

Todas as operações que envolvem a manipulação dos recursos virtuais realizadas durante implantação e otimização da *Cloud Slice* utilizam o módulo *Unified API*. A *Unified API* consiste dos componentes *Computing*, *Storage*, *Networking* e *Monitoring*, descritos a seguir. O componente *Computing* é responsável pelas funcionalidades das máquinas virtuais, como criação e migração, e também das imagens dos sistemas operacionais que são instaladas nas máquinas virtuais. O componente *Storage* lida com a alocação de volumes de armazenamento virtual. O componente *Networking* implementa interfaces para criação de links virtuais e roteadores virtuais. Existe um último componente, que é o *Monitoring*. Ele é responsável por configurar a infraestrutura de monitoração no exato momento em que uma *Cloud Slice* é implantada, além disso, implementa uma abstração de evento, que define alarmes para serem acionados a partir da infraestrutura de monitoração e utilizados pelo componente *Event Manager*.

As abstrações dos dispositivos virtuais são implementadas por um conjunto de *drivers* disponíveis no módulo localizado na parte inferior da arquitetura. Esses *drivers* são para tecnologias específicas e possuem dois papéis: abstraem a complexidade de configuração de parâmetros e protocolos de comunicação e deixam a plataforma do Aurora mais portável. Atualmente são utilizadas a *Libvirt*[16] para os componentes *Computing* e *Storage*, *Openflow* e *OpenVSwitch*[17] para *Networking*, e por último, *FlexACMS* e *Nagios* para o componente *Monitoring*.

O núcleo da plataforma Aurora é escrito na linguagem *Python* e implementado como uma aplicação *Web*, utilizando o *framework Django* [18].

4. Gerenciador Aurora Robusto

Esta seção descreve a plataforma robusta para gerência de recursos em nuvens IaaS baseada no gerenciador Aurora.

Concretamente, este trabalho tem como objetivo permitir a operação contínua do próprio ambiente de nuvem. A estratégia adotada é baseada na replicação do Aurora, mais especificamente do próprio gerenciador da nuvem. A replicação do Aurora é realizada em duas partes. A primeira parte é através da replicação do banco de dados MySQL, que é utilizado pelo Aurora para armazenar as informações de gerenciamento. Estas informações de gerenciamento são as seguintes: os *hosts*, com suas informações de endereço IP, nome, entre outras, a lista de máquinas virtuais instaladas nos *hosts*, roteadores, *switches*, a lista de programas de otimização e implantação das *Cloud Slices*, entre outras informações. A segunda parte é através da replicação de diretórios, onde o *rsync* [Tridgell 1999] é utilizado. Esta replicação de diretórios é necessária, pois as imagens utilizadas para criação de

[16]http://www.libvirt.org
[17]http://openvswitch.org/
[18]https://www.djangoproject.com

novas máquinas virtuais, os programas de implantação e otimização, além das métricas, ficam armazenados em diretórios, e não no banco de dados.

A figura 2 ilustra o funcionamento do Aurora para múltiplos *datacenters* com a replicação ativada. Nela existe o administrador da nuvem que acessa o sistema através de um único IP, mas ele pode acessar individualmente qualquer uma das três instâncias de Aurora. O acesso através de um único IP foi implementado utilizando-se o balanceador de carga *HAProxy*[19], que, quando recebe um acesso, redireciona o acesso para um dos três Auroras. Isto é, quando ocorrerem múltiplos acessos para o mesmo *datacenter*, o *HAProxy* redistribui os acessos entre os Auroras presentes no mesmo ambiente replicado, utilizando, por padrão, o algoritmo de balanceamento *round-robin*. No caso de múltiplos *datacenters*, é interessante utilizar o balanceamento por origem de comunicação, assim sempre um cliente acessa um Aurora específico, e somente se não houver mais Auroras replicados neste *datacenter*, o cliente acessa o Aurora do outro *datacenter*. O *HAProxy* também foi implementado com redundância de seus serviços, neste caso se um dos servidores do *HAProxy* falhar, o outro servidor consegue assumir e fazer o balanceamento dos acessos. A identificação de falha em um dos servidores do balanceador é feita pelo *Keepalived*[20].

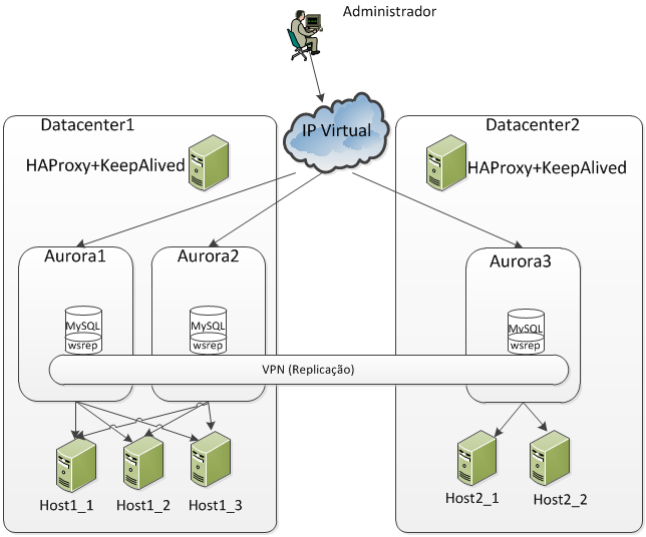


Figura 2. Arquitetura do Aurora em 2 *datacenters* com a replicação ativada.

Na figura, a replicação está ativada, então, cada instância do Aurora possui as informações das outras instâncias. Isto ocorre pois a cada inserção feita pelo administrador, por exemplo, a inserção de um novo *host* em uma instância do Aurora, ocorre a replicação para as outras instâncias. Todos os Auroras possuem acesso aos cinco *hosts*, *Host1_1*, *Host1_2*, *Host1_3*, *Host2_1* e *Host2_2*. Para a replicação funcionar é necessária a configuração de uma VPN (*Virtual Private Network*) entre os *datacenters*, pois é necessário manter um endereçamento de rede em que todos Auroras consigam se comunicar, além da segurança fornecida. Este cenário de múltiplos *datacenters* foi simulado forçando atrasos no enlace de comunicação entre eles. Esta simulação é descrita na próxima seção.

[19]http://www.haproxy.org/
[20]http://www.keepalived.org/

Neste trabalho, a abordagem de replicação escolhida foi a *multi-master*, pois permite a escrita e leitura de dados em qualquer instância independente do Aurora, mantendo os dados replicados e consistentes em todas as instâncias. A ideia principal de utilizar esta abordagem é de manter todos Auroras ativos e independentes, mas possuindo as informações dos outros, para prover a disponibilidade dos dados no caso em que ocorram falhas.

A replicação do banco de dados foi implementada através do *Galera Cluster*, que é um *plugin* do MySQL e é instalado juntamente com a plataforma Aurora. O Galera utiliza a abordagem de replicação *multi-master*, que é a abordagem de replicação escolhida para este trabalho.

A replicação do Aurora só é ativada quando o usuário habilita esta função, numa nova seção da interface gráfica do Aurora, chamada de *Aurora Cluster*. Esta seção contém a lista de Auroras participantes do *cluster*, com o estado (falho/não falho) de cada Aurora, assim como uma opção de ativação da função de replicação. Na ativação da função de replicação, o usuário é redirecionado para uma nova página informando sobre ativação da função, isto ocorre pois, para viabilizar a replicação, o serviço de *mysql* deve ser reiniciado para ser aplicado um novo arquivo de configuração do MySQL, com a informação do nome do *cluster*, endereços IP dos nós pertencentes ao *cluster*, entre outras informações. Na seção em que são listados os Auroras, o usuário deve cadastrar, em cada Aurora, quais são os Auroras participantes do *cluster* pois, assim, o sistema possuirá as informações dos participantes e também consegue ativar o serviço de replicação do Galera.

A solução proposta também possui a função de monitoramento dos nós participantes do *cluster* de Auroras. O monitoramento é feito através da ativação de um comando de notificação do Galera, que executa toda vez que existe uma alteração no *cluster* ou no estado do nó. Este comando então notifica alterações do estado dos nós para o Aurora. Quando é detectado que um Aurora falhou e não pertence mais ao *cluster*, é enviada uma notificação por *email* ao administrador com a informação do endereço IP do Aurora que falhou.

Vale ressaltar que por segurança, toda a comunicação entre os *hosts* e o Aurora é criptografada utilizando conexões via *ssh*. As chaves *ssh* já devem estar instaladas em todos os *hosts*, para que outra instância do Aurora consiga gerenciar qualquer *host* físico. Para a replicação que utiliza o *rsync*, também é necessário copiar estas chaves *ssh* para as outras instâncias de Auroras.

5. Experimentos

Nesta seção são apresentados os experimentos realizados para avaliação do gerenciador de nuvem Aurora robusto. O primeiro experimento mede o tempo necessário para que uma nova instância do Aurora seja incorporada a um *cluster* de Auroras já existente. O segundo experimento mede a latência de replicação de dados, tanto para as informações do banco de dados quanto para as imagens, programas e métricas. A latência de replicação é medida tanto em um *datacenter* quanto em dois *datacenters*. No cenário com um *datacenter*, o experimento foi realizado com dois Auroras em um primeiro momento e três Auroras em seguida. No cenário com dois *datacenters*, há um Aurora em cada *datacenter*. Este cenário, com dois *datacenters*, simula um enlace remoto. O terceiro experimento mede o tempo de detecção e recuperação de falhas, e tem o intuito de analisar a disponibilidade

do sistema. Para todos os experimentos assume-se que o canal de comunicação entre os Auroras é confiável. Os Auroras utilizados são instalados em máquinas virtuais com Sistema Operacional Ubuntu 14.04.02 LTS, que possuem 512MB de memória RAM. Os canais de comunicação remotos foram simulados oscilando atrasos progressivos no canal.

O primeiro experimento realizado mede o tempo necessário para que uma nova instância do Aurora seja incorporada a um *cluster* de dois Auroras. Quando uma nova instância é inserida num *cluster*, ela deve se comunicar com outra instância e receber todos os dados relevantes. Os tempos deste experimento foram medidos no relógio local e estão descritos no gráfico 3. Existe um tempo padrão de entrada de uma nova instância no *cluster*, que é de 15 a 18 segundos. O tamanho do banco de dados foi aumentado gradativamente. Este tamanho é representado pelo número de máquinas virtuais configuradas no Aurora. Na figura, é possível observar que o número de máquinas virtuais influencia diretamente no tempo de entrada do terceiro Aurora no *cluster*. Com somente uma máquina virtual configurada, o tempo de entrada de uma instância foi de 20 segundos. Com o número de 5000 máquinas virtuais configuradas, o tempo necessário para que uma nova instância fosse incorporada no *cluster* foi de 40 segundos. Com esses números, é possível concluir que, para um aumento no número de máquinas virtuais de 5000 vezes, o tempo de entrada de um novo Aurora apenas dobrou.

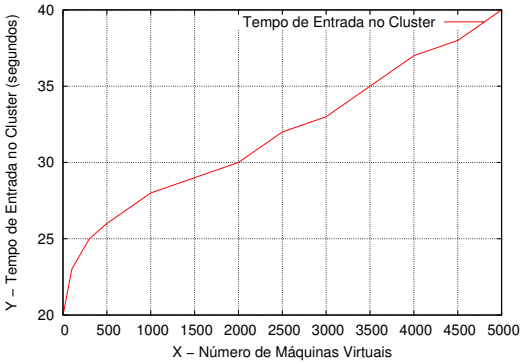


Figura 3. Gráfico de tempo de inserção de um novo Aurora.

O segundo experimento tem o propósito de medir a latência para replicação de dados e foi dividido em três partes. Na primeira parte, em um primeiro momento, é realizada a medição da latência de replicação dos dados mantidos pelo banco de dados MySQL, pela simples adição de configuração em um dos Auroras. Em sequência, é medida a latência de replicação das imagens das máquinas virtuais, após a inserção de uma imagem no Aurora. Na segunda parte, a latência de replicação foi medida com um aumento progressivo da carga de inserção de dados no Aurora. Na terceira parte, a latência de replicação foi medida pela adição de configuração em um dos Auroras, em um cenário com atrasos no canal de comunição entre as instâncias do Aurora, simulando um enlace remoto.

Na primeira parte do segundo experimento, a medição da latência foi realizada através dos *logs* do banco de dados. A imagem que foi inserida no Aurora possui um tamanho de 2,5MB. A medição da latência da replicação da imagem foi realizada com a utilização de relógios locais. Nesta primeira parte, o experimento foi executado em um cenário com dois gerenciadores Aurora em um primeiro momento, e três Auroras no

momento seguinte. O experimento foi executado em um único *datacenter*. No cenário com dois Auroras, o tempo de replicação dos dados mantidos no MySQL é de milésimos de segundos, com média de 21,63 ms, após 30 execuções. Na replicação das imagens, a latência média foi de 4,7 segundos, também para 30 execuções. No segundo cenário, com três Auroras, a replicação dos dados do MySQL também teve latência de milésimos de segundos, a média foi um pouco maior do que o cenário com dois Auroras, de 36,95 ms. A replicação das imagens teve tempo médio de 5,96 segundos. A tabela 1, resume o experimento.

Tabela 1. Experimentos de Latência de Replicação

Tipo de Replicação/Cenário	2 Auroras		3 Auroras	
	Média	Desvio Padrão	Média	Desvio Padrão
Replicação MySQL	21,63 ms	11,45 ms	36,95 ms	17,33 ms
Replicação Rsync	4,7 s	0,95 s	5,96 s	1,56 s

Na segunda parte do segundo experimento, a variação de latência foi medida com um aumento da carga da inserção de dados no banco de dados. O experimento também foi executado em ambientes com duas e três instâncias do Aurora. Além do tempo de replicação, a utilização de CPU e rede também foram medidas. O uso de memória é constante na máquina virtual. Na máquina com 512MB de memória, a utilização é de 95%. A inserção de dados foi executada através da configuração de novas máquinas virtuais através da importação de um arquivo *xml*.

A figura 4 ilustra o tempo de replicação em relação a um aumento progressivo de máquinas virtuais gerenciadas pelo Aurora. Existem duas linhas, uma representando o experimento com dois Auroras e outra representando o experimento com três Auroras. Nota-se que o tempo de replicação aumenta conforme aumenta-se a carga de inserção de máquinas virtuais e também quando existem mais gerenciadores Aurora no *cluster*, sendo exatamente este o comportamento esperado. Com a inserção de dados de 4000 máquinas virtuais, o tempo de replicação com um *cluster* de duas instâncias do Aurora foi de aproximadamente 750 ms e com um *cluster* de três instâncias foi de aproximadamente 1290 ms.

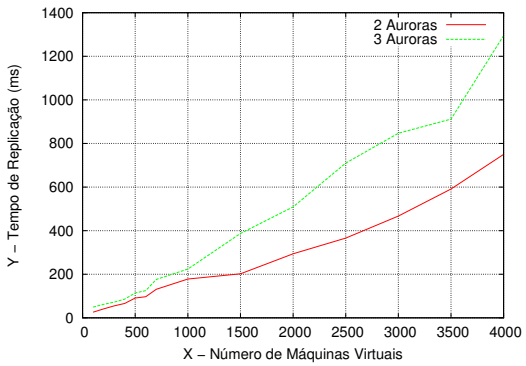
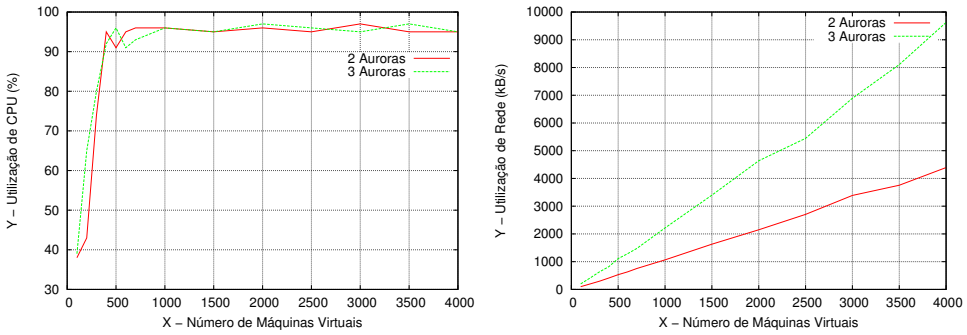


Figura 4. Gráfico de tempo de replicação conforme aumento do banco de dados.

As figuras 5(a) e 5(b), ilustram a utilização de CPU durante o processo de inserção de máquinas virtuais e a utilização da interface de rede, respectivamente. No gráfico de utilização de CPU é possível notar que a partir da inserção de 400 máquinas virtuais, a utilização de CPU se manteve constante, com mais de 92%, tanto para um *cluster* de duas instâncias do Aurora quanto para um *cluster* de três instâncias. No gráfico de utilização

da rede, conforme o número de inserções de máquinas virtuais aumenta, a utilização de rede também aumenta, o que é um comportamento normal, pois existem mais dados para serem replicados entre os membros do *cluster* de Auroras. A utilização de rede sempre foi pelo menos o dobro em um *cluster* com três instâncias do Aurora em relação a um *cluster* com duas intâncias. Isto acontece pois a replicação dos dados com três instâncias sempre será duplicada, uma vez que a instância que recebe os dados deve replicar os mesmos dados para as outras duas instâncias. Após a inserção de 4000 máquinas virtuais, a utilização de rede com três instâncias do Aurora foi de 9600 kB/s e com duas instâncias foi de 4360 kB/s.



(a) Utilização de CPU . (b) Utilização de Rede.

Figura 5. Avaliação de utilização de CPU e rede durante a replicação.

Na terceira parte do segundo experimento foi realizada a simulação de dois *datacenters* remotos, com um Aurora em cada um dos *datacenters*. Para simular a comunicação entre os dois *datacenters*, foram adicionados atrasos no enlace entre os dois Auroras, através do aplicativo *tc (Traffic Control)*. A inserção dos dados para replicação do banco de dados foi realizada através da simples adição de configuração no Aurora. Os tempos de replicação foram obtidos através dos *logs* do MySQL.

A figura 6 mostra a relação entre a latência de replicação e o atraso gradual no canal de comunicação entre dois Auroras. O principal objetivo deste experimento é conseguir verificar o limite do sistema para o caso de dois *datacenters* remotos. O limite do sistema acontece com um atraso de 4000 ms no enlace que conecta os dois *datacenters* simulados. Com esse atraso, a interface web do Aurora começa a apresentar lentidão e não consegue progredir com sua execução. Acontece este travamento, porque com a lentidão, o processo de replicação do Galera não consegue escrever a transação no banco de dados (não consegue obter o *lock*) e ocorre o *timeout*. Na figura, também é possível notar que a partir do atraso de 1000 ms, a latência de replicação começa a aumentar mais rapidamente, isto acontece, pois o tempo para o Galera adquirir o *lock* para escrever a transação no banco de dados é maior.

Foram também realizados testes, aumentando-se a quantidade de memória da máquina virtual do Aurora, para 1024MB. A ideia destes testes é verificar se, com um aumento de memória, o tempo de replicação diminui. Com esse aumento aconteceu uma pequena melhora nos tempos de replicação do experimento e não ocorreu o travamento do sistema com 4000 ms, mas com um atraso de 5000 ms.

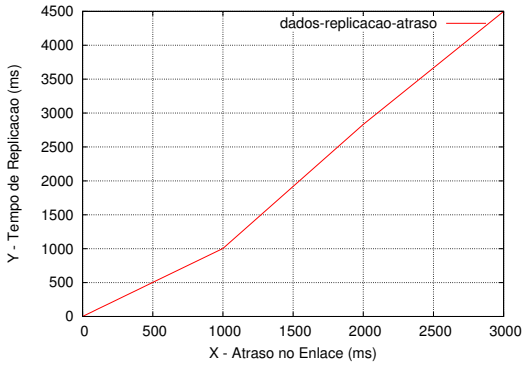


Figura 6. Gráfico de tempo de replicação com atraso no enlace entre dois *data-centers*.

A tabela 2 mostra o tempo médio de acesso a uma instância Aurora, com o aumento progressivo de atraso no enlace. Foram executadas 50 capturas do tempo de RTT (*Round-Trip Time*) para verificar o tempo de acesso e são mostradas as médias obtidas. A solução proposta funciona corretamente para atrasos de até 3 segundos, demonstrando sua viabilidade em redes reais.

Tabela 2. Experimentos de Tempo de Acesso/ Tempo de Replicação - Com atraso no Enlace

Atraso (ms)	Latência de Acesso (ms)	Latência de Replicação (ms)
0	0.485	0.521
10	10.834	10.914
50	50.753	50.824
100	100.814	101.894
500	500.768	502.681
1000	1000.883	1002.478
2000	2000.740	2832.474
3000	3000.905	4498.797

O terceiro experimento tem o intuito de analisar a disponibilidade do sistema. O tempo de detecção de falha e o tempo de recuperação após uma falha são medidos neste experimento. O MTBF (*Mean Time Between Failures*), ou período médio entre falhas, é usado para calcular a disponibilidade. O tempo de detecção de falha é o tempo configurado no *HAProxy*. Na configuração utilizada neste trabalho, a verificação de falha no Aurora acontece a cada 2000 ms e, se acontecem dois resultados de falha, o serviço é considerado falho. Assim, o tempo de detecção é de 4000ms (4s). A quantidade de resultados falhos para considerar que um servidor está falho, além dos intervalos de verificação, são configuráveis. Se o serviço é considerado falho, então qualquer nova tentativa de acesso ao Aurora, é realizada somente no Aurora que está ativo.

Existe o tempo de detecção do próprio Galera. Se um processo do MySQL em uma máquina do *cluster* sofre falha, a detecção ocorre em milésimos de segundos. Analisando-se *logs* obtidos durante a execução dos experimentos, a detecção ocorre em até 15 ms. No caso em que ocorre uma falha de comunicação com outro Aurora, o Galera considera o nó como suspeito em 5 segundos, mas somente o retira do *cluster*, se não obtiver resposta no tempo de 15 segundos. Vale ressaltar que estes parâmetros são configuráveis.

No terceiro experimento também foi medido o tempo de recuperação de um Au-

rora após uma falha. A falha provocada é a parada do serviço MySQL e a recuperação é o retorno deste serviço. Os experimentos foram realizados com duas e três instâncias de Auroras. O tempo de inicialização do serviço no cenário com dois Auroras foi de 16 segundos. No cenário com três Auroras, o tempo de inicialização foi de 18 segundos. Os tempos deste experimento também foram medidos através de um relógio local.

Na tabela 3, são mostrados os valores de disponibilidade do sistema, calculados a partir do MTBF. No cálculo do MTBF é considerado que o período de indisponibilidade do sistema é o intervalo de monitoramento do *HAProxy*. Nos cálculos, o tempo do intervalo de monitoramento é representado por im e a quantidade de verificações para considerar que um servidor está falho vf. Outra medida representada é o tempo de MTBF $tmtbf$, que representa o tempo em que são consideradas as falhas, por exemplo, o $tmtbf$ sendo 30 minutos significa que considera-se o tempo de 30 minutos entre duas falhas consecutivas. A quantidade de vf e $tmtbf$ foram variadas nos cálculos. O cálculo usado para a disponibilidade é o seguinte:

$$disponibilidade = \frac{(tmtbf(segundos) - (im * vf)) * 100)}{tmtbf}$$

Tabela 3. Tempo de disponibilidade.

Tempo MTBF	Intervalo de Monitoramento (im)	Quantidade de Verificações (vf)	Disponibilidade (%)
1 hora	2 segundos	1	99,94
		2	99,88
		5	99,72
2 horas	2 segundos	1	99,97
		2	99,94
		5	99,86
4 horas	2 segundos	1	99,98
		2	99,97
		5	99,93

Nota-se na tabela, que apesar da disponibilidade ser sempre menor quando o parâmetro vf é igual a 5, a precisão da detecção é maior, com probabilidade menor de gerar um falso-positivo. Um exemplo é: se acontecer uma falha em uma instância do Aurora e ele permanecer indisponível por 5 segundos e logo retornar, se o vf conter valor de 5, a instância não será considerada falha. No entanto, se o vf for igual a 2, a instância será considerada falha, gerando um falso-positivo. Este experimento demonstra que os parâmetros de detecção de falhas devem ser bem analisados, tanto o intervalo de monitoramento, quanto quantidade de resultados falhos para considerar que um servidor está falho. A análise deve ser tomada, considerando o aspecto de disponibilidade, e/ou precisão na detecção de uma falha.

6. Conclusão

Este trabalho propôs a implementação de uma plataforma robusta para gerência de recursos em nuvens IaaS (*Infrastructure as a Service*). A solução foi implementada no gerenciador de nuvens *Aurora Cloud Manager*. A estratégia é baseada na replicação do Aurora, mais especificamente o próprio gerenciador da nuvem. A principal contribuição deste trabalho é oferecer, no próprio gerenciador de nuvem, a *feature* de alta disponibilidade do próprio gerenciador, com a opção de ativação da replicação em sua interface e monitoramento das outras instâncias replicadas.

O desempenho e a robustez da plataforma foram avaliados experimentalmente. Foram realizados experimentos para a medição do tempo necessário para que uma nova

instância do Aurora seja incorporada a um *cluster* de Auroras já existente; medição da latência de replicação de dados, tanto para as informações do banco de dados quanto para as imagens, programas e métricas. A latência de replicação foi medida tanto em um *datacenter* quanto em dois *datacenters* remotos. E, por último, a medição do tempo de detecção e recuperação de falhas, com o intuito de analisar a disponibilidade do sistema. Os resultados mostraram que o gerenciador se tornou robusto e capaz de ser executado com múltiplas instâncias replicadas.

Entre os trabalhos futuros, pretende-se implementar o cenário com múltiplos *datacenters* remotos utilizando federações de nuvens para o compartilhamento de recursos.

Referências

Arean, O. (2013). Disaster Recovery in The Cloud . *Network Security*, 2013(9):5 – 7.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A View of Cloud Computing. *Communications of The ACM*, 53(4):50–58.

Barbosa de Carvalho, M., Pereira Esteves, R., da Cunha Rodrigues, G., Zambenedetti Granville, L., and Rockenbach Tarouco, L. (2013). A Cloud Monitoring Framework for Self-Configured Monitoring Slices Based on Multiple Tools. In *9th International Conference on Network and Service Management (CNSM)*.

Charron-Bost, B., Pedone, F., and Schiper, A., editors (2010). *Replication: Theory and Practice*. Springer-Verlag, Berlin, Heidelberg.

Jhawar, R. and Piuri, V. (2012). Fault Tolerance Management in IaaS Clouds. In *2012 IEEE First AESS European Conference on Satellite Telecommunications (ESTEL)*, pages 1–6.

Kholghi, M. A. K., Abdullah, A., Latip, R., Subramaniam, S., and Othman, M. (2014). Disaster Recovery in Cloud Computing: A Survey. *Computer and Information Science*, 7(4):39–54.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Mell, P. and Grance, T. (2011). The NIST Definition of Cloud Computing.

Tridgell, A. (1999). *Efficient Algorithms for Sorting and Synchronization*. PhD thesis, Australian National University Canberra.

Tsakalozos, K., Roussopoulos, M., and Delis, A. (2011). VM Placement in non-Homogeneous IaaS-Clouds. In Kappel, G., Maamar, Z., and Motahari-Nezhad, H., editors, *Service-Oriented Computing*, volume 7084 of *Lecture Notes in Computer Science*, pages 172–187. Springer Berlin Heidelberg.

Wickboldt, J. A., Esteves, R. P., de Carvalho, M. B., and Granville, L. Z. (2014). Resource Management in IaaS Cloud Platforms Made Flexible Through Programmability . *Computer Networks*, 68(0):54 – 70. Communications and Networking in the Cloud.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud Computing: State-of-The-Art and Research Challenges. *Journal of Internet Services and Applications*, 1(1):7–18.

Trilha Principal do SBRC 2017
Sessão Técnica 2
Redes Definidas por Software I

Um Protocolo Simples e Eficiente para Atualização Consistente de Políticas em Redes Definidas por Software com Controle Distribuído*

Diogo M. F. Mattos[1], Otto Carlos M. B. Duarte[1] e Guy Pujolle[2]

[1]Grupo de Teleinformática e Automação
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

[2]Laboratoire d'Informatique de Paris 6
Sorbonne Universities, UPMC Univ Paris 06
Paris, France

Resumo. *Novas políticas são instaladas na rede a todo instante. Nas Redes Definidas por Software, os diversos controladores distribuídos têm que instalar as novas políticas de forma consistente, para não haver riscos de a rede passar por estados de configuração transitórios e inesperados, que comprometam a segurança ou mesmo a operação. Este artigo propõe um protocolo de consistência para serializar as atualizações de políticas e para compor as políticas evitando conflitos. As principais contribuições são: (i) um protocolo de consistência para serializar a atualização de políticas; (ii) uma interface de consenso para os controladores acordarem sobre a versão mais recente da configuração da rede; e (iii) um algoritmo para verificar se a nova política é uma atualização, um refinamento, ou se entra em conflito com outras políticas já instaladas. Através de verificação formal, mostra-se que o protocolo de consistência proposto garante uma ordem global para todas as atualizações de políticas e que o algoritmo proposto compõe corretamente todas as políticas. A simulação da proposta em uma topologia de rede real mostra que a atualização distribuída de políticas é consistente por pacote e apresenta uma baixa sobrecarga de mensagens de controle.*

Abstract. *New policies are constantly installed on the network. In Software Defined Networks, the various distributed controllers have to install the new policies consistently for assuring that there is no risks of network experiences transient and unexpected configuration states, which compromise the safety and the operation. In this paper, we propose a consistency protocol for serializing policy updates and for composing policies, avoiding conflicts. The main contributions are three-fold: (i) a consistency protocol for serializing policy updates; (ii) a consensus interface for enabling controllers to agree on the latest version of the network configuration; and (iii) an algorithm to verify that the new policy is an update, a refinement, or if it conflicts with other policies that have already been installed. Through formal verification, we show that the proposed consistency protocol ensures global order for all policy updates and that the proposed algorithm correctly composes all policies. The simulation of the proposal in a real network topology shows that the distributed policy update is per-consistent packet and it has a low overhead of control messages.*

*Este trabalho foi realizado com recursos do CNPq, CAPES e FAPERJ.

1. Introdução

O gerenciamento de redes envolve a definição contínua de políticas de rede que incluem a engenharia de tráfego e o encadeamento de funções de rede [Mattos et al. 2016a, Han et al. 2015, Reitblatt et al. 2012]. O paradigma de Redes Definidas por Software (*Software Defined Networking* - SDN) simplifica o gerenciamento, uma vez que separa o plano de controle, logicamente centralizado, do plano de dados distribuído [Levin et al. 2012]. Aplicações de rede, localizadas no plano de controle, acessam uma visão global consistente da rede. Isso permite definir políticas de alto nível que codificam o comportamento esperado da rede [Canini et al. 2015]. As políticas são traduzidas para o plano de dados, no qual se configura o encaminhamento e tratamento dos pacotes. Em SDN, as regras de encaminhamento são expressas por configurações de fluxo nas tabelas dos comutadores em execução no plano de dados.

O controle logicamente centralizado é o principal pilar do paradigma SDN, embora a realização do controlador de rede como um servidor centralizado implique desafios para a segurança, o desempenho e a escalabilidade da rede [Mattos et al. 2016a]. Portanto, o controle e a consequente manipulação das atualizações de política em SDN são um desafio de computação distribuída. O bom funcionamento da rede depende da coordenação e do tratamento consistente das atualizações de políticas que chegam simultaneamente aos controladores, além da composição da interação entre todas as políticas aplicadas na rede [Canini et al. 2015, Reitblatt et al. 2012].

Este artigo propõe um protocolo simples e eficiente para a serialização[1] da instalação de atualizações de políticas em Rede Definidas por Software com um plano de controle distribuído. A ideia principal é garantir o consenso entre os controladores quanto à aplicação de uma nova política sobre a rede. Quando atualizações de política chegam concomitantemente a diferentes controladores, esses devem concordar sobre a ordem de instalação de todas as atualizações requisitadas e, também, se a nova política gera conflito com as demais. Por isso, as principais contribuições deste artigo são três: (i) um protocolo simples de consistência, que serializa a instalação de atualizações de políticas simultâneas lançadas por diferentes controladores; (ii) uma interface de consenso abstrata, na qual os controladores acordam sobre a versão mais atual da configuração da rede; e (iii) um algoritmo simples para verificar se uma nova política é uma atualização, um refinamento, ou se está em conflito com outras políticas já instaladas na rede.

O restante do artigo está organizado da seguinte forma. Na Seção 2, apresentam-se os trabalhos relacionados. A Seção 3 discute os desafios da atualização das políticas em Redes Definidas por Software com um plano de controle distribuído. Na Seção 4, propõe-se o protocolo de consistência para atualizações de políticas em Redes Definidas por Software com controle distribuído. As simulações e os resultados são apresentados e discutidos na Seção 5. A Seção 6 conclui o artigo.

2. Trabalhos Relacionados

Reitblatt *et al.* propõem o esquema de atualização de duas fases (*Two-Phase Update*), em que aplicam o modelo de consistência por pacote para a atualização consistente

[1]No contexto deste artigo, serialização refere-se ao ordenamento de ações na rede sem que haja sobreposição temporal entre ações.

de regras nos comutadores em SDN com controle centralizado [Reitblatt et al. 2012]. Esse esquema de atualização de duas fases insere etiquetas nos pacotes quando entram na rede, indicando a marcação da versão mais recente da configuração. Durante o encaminhamento do pacote na rede, ele é sempre processado de acordo com a versão da configuração da rede marcada na etiqueta, o que evita a manipulação de pacotes por duas versões distintas da configuração da rede. No processo de atualização, a primeira fase acrescenta novas regras nos comutadores do núcleo da rede, que aplicam a nova política aos pacotes marcados com a nova etiqueta de versão da configuração. A segunda fase atua nas portas de entrada dos comutadores de borda. Nessa fase, o sistema atualiza as regras sobre os comutadores de ingresso para marcar os pacotes entrantes com a nova versão.

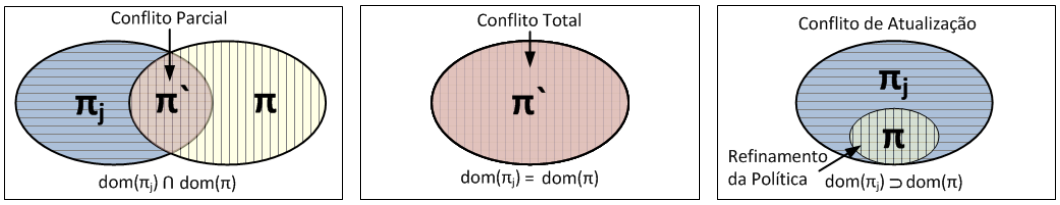
Canini *et al.* estendem a proposta de atualização de duas fases a uma Rede Definida por Software com controle distribuído [Canini et al. 2015]. Canini *et al.* introduzem o problema de Composição Consistente de Políticas (*Consistent Policy Composition* - CPC), no qual eles definem que para aceitar atualizações de políticas de uma forma consistente em uma rede com controle distribuído é necessário compor todas as políticas em uma configuração de rede única e sem conflitos. As políticas aceitas não podem entrar em conflito com as políticas já implantadas ou com outras instalações de políticas concomitantes. Os autores propõem uma interface transacional para a busca de conflitos entre as políticas e, após, a aceitação simples das políticas que não entrem em conflito com as demais. Embora Canini *et al.* resolvam o problema de Composição Consistente de Políticas e alcancem a complexidade ideal de etiquetas, a proposta assume a pré-existência de uma abstração consenso entre os controladores.

Mattos *et al.* propõem que a atualização das regras na rede seja feita na ordem inversa dos fluxos, garantindo assim a consistência e evitando a necessidade de marcação de pacotes com etiquetas de versão [Mattos et al. 2016b, Mattos e Duarte 2015]. McGeer propõe realizar o armazenamento temporário dos pacotes em trânsito no controlador durante a atualização da rede. Após a atualização, os pacotes são reinjetados na rede [McGeer 2012]. Katta *et al.* propõem realizar a atualização em rodadas. Em cada rodada, eles executam uma atualização de duas fases em um subconjunto de fluxos. A ideia principal é remover o antigo conjunto de regras após cada rodada, liberando a memória dos comutadores [Katta et al. 2013]. McClurg *et al.* propõem um algoritmo para procurar automaticamente uma ordem para atualizar a configuração de rede, em que sejam garantidas as propriedades invariantes da rede em todos os estados de configuração transitórios [McClurg et al. 2015]. Essas propostas, no entanto, focam em Redes Definidas por Software com um controle centralizado.

No caso de nós distribuídos, a ideia mais simples de consenso é decidir se uma transação deve ser confirmada ou não [Gray e Lamport 2006]. Um protocolo simples para alcançar um consenso é o protocolo *Two-Phase Commit*, ou efetivação em duas fases, que utiliza uma fase para propor uma transação e, depois de todos os nós confirmarem que concordam com a transação, uma segunda fase envia o comando de efetivação a todos os nós. No entanto, o *Two-Phase Commit* pode gerar uma situação de impasse, caso o nó que inicia o protocolo falhe. Uma solução para a resolução do impasse é o protocolo de efetivação de três fases (*Three-Phase Commit*) que adiciona uma fase extra entre a votação e a efetivação da transação, para facilitar a recuperação do estado da transação no caso de falha do nó iniciador.

3. O Problema de Composição Consistente de Políticas

O plano de controle logicamente centralizado consiste em uma abstração de uma visão rede global compartilhada por todos os controladores da rede [Mattos et al. 2016a]. Assim, todos os controladores têm acesso a uma interface de consenso para atualizar sua visão de rede global. Em um cenário de controle distribuído, diferentes controladores podem realizar requisições atualizações de políticas concomitantes na rede. A fim de aplicar de forma consistente as atualizações na rede, cada controlador tem que estar consciente sobre a ordem de instalação das atualizações e, também, se uma requisição de uma nova política entra em conflito com as demais. Desta forma, Canini *et al.* definem o problema da Composição Consistente de Política (*Consistent Policy Composition* - CPC) [Canini et al. 2015], que consiste em consolidar as políticas de rede que chegam aos controladores concomitantemente em uma configuração da rede única, consistente e global, na qual não haja conflito entre diferentes políticas.



(a) Conflito parcial entre π e π_j. (b) Conflito total entre π e π_j. (c) Conflito de atualização entre π e π_j.

Figura 1. Casos possíveis de conflitos entre a nova política π com uma política já instalada π_j no espaço de fluxos S: a) A solução para um conflito parcial é a criação de um subconjunto de políticas π'; b) A solução para um conflito total é a avaliação das duas políticas e a consequente instalação de uma nova política π' que considere os casos em conflito; c) π é o refinamento de uma política π_j ou uma atualização explícita. O conflito de atualização é trivialmente resolvido aplicando-se a política π.

Considerando-se o problema de atualização de políticas em um ambiente de controle distribuído, identificam-se duas propriedades que devem ser satisfeitas: a consistência e a composição. A instalação consistente de políticas consiste em agendar requisições de atualização de políticas e garantir que é possível estabelecer uma ordem global entre todos os pedidos programados. Em um segundo momento, considera-se o problema de composição de políticas. Assim, o problema composição lida com a aceitação, ou rejeição, de uma nova política, ou uma atualização de política, considerando as políticas que já foram aplicadas à rede. Portanto, dois subproblemas são definidos: serialização das requisições e composição de políticas.

O problema de serialização de requisições consiste em definir uma ordem global consistente entre todas as requisições concomitantes. Seja H a história da rede, isto é, o conjunto de todos os eventos que acontecem na rede. Uma relação parcial de ordem nos eventos da história H é definida como $<_H$. Uma requisição req precede outra requisição req' na história H, representada por $req <_H req'$, se a reposta de req aparece antes da chamada de req' em H [Canini et al. 2015, Mattos e Duarte 2015]. Se duas requisições não estão relacionadas em uma ordem de precedência, diz-se que ambas são requisições concomitantes. De forma similar, um evento de rede ev, como a chegada de um novo

pacote, precede uma requisição req em H, representado por $ev <_H req$, se ev ocorre antes da chamada de req em H. Ademais, o evento ev sucede req em H, $req <_H ev$, se ev ocorre após a resposta à req. Um evento ev é concorrente com a requisição req se $ev \not<_H req$ e $req \not<_H ev$. A história H é sequencial se não existe duas requisições concorrentes nem um evento concorrente com uma requisição.

Seja H_{c_i} a história local controlador c_i uma subsequência de H, que consiste de todos os eventos e requisições que ocorrem em c_i. Localmente, verifica-se que toda e qualquer história H_{c_i} é sequencial, já que um controlador aceita uma nova requisição se, e somente se, não existir uma requisição prévia sem reposta, nem um evento sendo tratado. A história H é consistente se a relação de precedência entre duas requisições na história local de um controlador é mantida em qualquer outra história local de outros controladores. Assim, uma requisição consistente mantém a propriedade

$$req <_{H_{c_i}} req' \rightarrow req <_{H_{c_j}} req', \tag{1}$$

$\forall \{req, req'\} \subset H_{c_i} | req <_H req'$ e $\forall c_j \in C$, onde C é o conjunto de todos os controladores da rede. Assim, considera-se que a ordem local de todos os controladores é consistente com a ordem global definida em H.

Embora a ordem global defina a serialização consistente das requisições, ainda é necessário definir corretamente a composição entre as novas políticas e as já aplicadas na rede. A fim de abordar a composição de políticas, considera-se que H deve respeitar as seguintes propriedades [Canini et al. 2015]:

- uma política é aceita com sucesso para ser adicionada à H se, e somente se, não entra em conflito com nenhuma outra política já aplicada à H;
- para cada evento de entrada de pacote ev na rede, o comportamento da rede é consistente com a composição de todas as políticas aplicadas com sucesso em H e que precedam o evento ev em H.

O conflito entre políticas surge quando duas políticas agem em conjuntos de fluxos de sobrepostos [Ferguson et al. 2012, Luo et al. 2015]. Políticas livres de conflitos são aquelas que possuem domínios completamente disjuntos[2]. Sejam π uma requisição de atualização de política, Π o conjunto de todas as políticas já instaladas na rede, e $dom(\pi)$ o domínio da política π, então as políticas não-conflitantes respeitam a propriedade

$$dom(\pi) \cap dom(\pi_j) = \emptyset, \forall\, \pi_j \in \Pi. \tag{2}$$

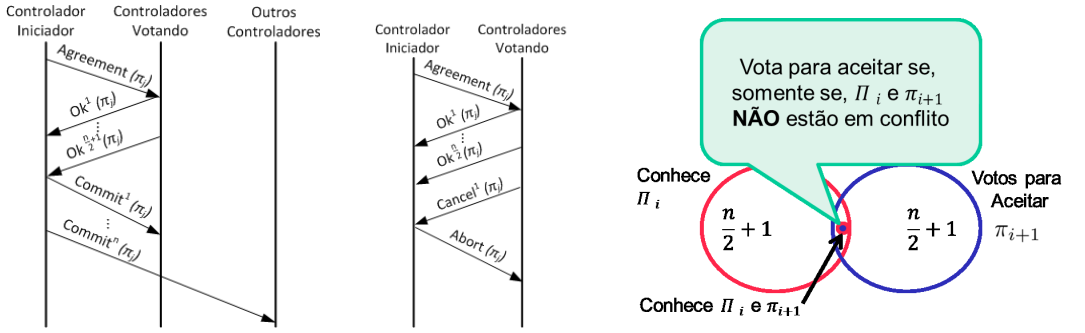
É possível identificar três tipos de conflito: conflito parcial, conflito total e conflito de atualização. O conflito parcial é quando o domínio de uma política sobrepõe o domínio de outra. Nesse caso, a composição das duas políticas pode levar à criação de regras de manipulação de pacotes para cada subconjunto, como mostrado na Figura 1(a). O conflito total ocorre quando duas políticas diferentes têm o mesmo domínio. Nesse caso, mostrado na Figura 1(b), a nova política substitui a anterior, ou a nova política é totalmente rejeitada. O terceiro caso é o conflito de atualização, Figura 1(c). De fato, o conflito de atualização

[2]O domínio de uma política é o conjunto de fluxos que a política afeta, o subespaço de fluxo (partição do *flowspace*) [Reitblatt et al. 2012, Ferguson et al. 2012].

ocorre quando uma nova versão da política anterior é lançada e, assim, a anterior deve ser explicitamente substituída por uma nova política com o mesmo domínio. Em um caso de instalação de uma política que é um refinamento da anterior, quando o domínio de nova política é subconjunto de uma política instalada anteriormente, a nova requisição deve ser tratada como um conflito de atualização.

4. O Protocolo de Consistência Proposto

A ideia principal da proposta é definir uma ordem global entre todas as requisições de atualização de política. A instalação de uma política na rede ocorre em três passos. O primeiro é o recebimento da requisição de atualização de política por um controlador. O segundo passo é o lançamento da política na rede, que consiste em votar a sua aplicabilidade e o número de ordem a ser assumido pela política. Se a atualização política é aceita por uma maioria de controladores, recebe, então, um número global de ordem. A partir de então, inicia-se o terceiro passo, em que a atualização de política é efetivada na rede seguindo o esquema de atualização de duas fases [Reitblatt et al. 2012]. A efetivação da atualização de política consiste na instalação da política nos comutadores, traduzindo a política de alto nível em regras de manipulação de pacotes. A proposta deste artigo concentra-se no primeiro e segundo passos.



(a) Instalação bem sucessidida da política π_j.

(b) Aborto da instalação da política π_j.

(c) Ilustação do algoritmo de verificação de conflitos.

Figura 2. O protocolo de consistência proposto. O controlador iniciador recebe uma requisição de atualização de políticas e começa a instalação da política π_j na rede. a) Todos $n/2+1$ controladores concordam em aceitar a instalação de π_j. O controlador iniciador envia as mensagens de `commit` a todos controladores na rede. b) Se ao menos um controlador discordar da instalação de π_j, o iniciador envia uma mensagem de `abort` para todos os controladores que estavam votando na aceitação da política. c) Algoritmo de verificação de conflitos garante que ao menos um nó dos votantes conhece todas as políticas já instaladas.

O protocolo de consistência proposto funciona da seguinte maneira. A requisição de atualização de política chega a um controlador[3]. Requisições podem chegar simultaneamente em diversos controladores. O controlador iniciador é aquele que recebe uma requisição, vinda de uma aplicação de controle, inicia a execução do protocolo e envia

[3]Uma requisição de atualização de política chega ao controlador através da *Northbound API* ou através da East/Westbound API. No primeiro caso, uma aplicação de controle gera a requisição a ser tratada pelo controlador. No segundo caso, o controlador recebe uma mensagem de `agreement` de outro controlador na rede.

mensagens de `agreement` para $n/2 + 1$ outros controladores, em que n é o número total de controladores na rede. A mensagem de `agreement` contém a requisição de atualização de política, assim como o domínio de atuação da política e o número de ordem proposto, dado pela ordem mais recente conhecida pelo iniciador acrescida de uma unidade. Quando um controlador recebe uma mensagem de `agreement`, duas mensagens de resposta são possíveis: `ok` e `cancel`. Se todos os $n/2 + 1$ controladores respondem com `ok`, o controlador iniciador então envia uma mensagem de `commit` para todos os controladores na rede, indicando que o processo de instalação pode ser realizado, como mostrado na Figura 2(a). Ao receber uma mensagem `commit`, o controlador verifica se o número de ordem na mensagem é compatível com a versão da próxima configuração de rede que ele espera receber. Em caso afirmativo, o controlador instala a nova política. No caso de a mensagem `commit` apresentar um número de ordem superior ao que os controlador estava esperando, o controlador sincroniza a sua base de dados de políticas instaladas como o controlador iniciador. No caso de o controlador iniciador receber uma mensagem `cancel` como resposta à sua mensagem de `agreement`, o controlador iniciador aborta a instalação da atualização de política e sincroniza sua base de políticas com o outro controlador que enviou a mensagem de `cancel`, como mostrado na Figura 2(b).

Execução do protocolo. Como mostrado na Figura 2(a), se $n/2+1$ controladores concordam em efetivar a atualização da política, a informação é propagada para todos os controladores e a atualização da política está instalada na rede muda de acordo com o esquema de atualização de duas fases (*Two-Phase Update*) [Reitblatt et al. 2012]. No entanto, falhas podem surgir durante a execução do protocolo proposto. Quatro casos de falha são tratados pelo protocolo: i) mais do que um controlador iniciam o protocolo ao mesmo tempo; ii) o controlador iniciador falha após o envio da mensagem de `agreement` para qualquer nó; iii) um nó falha depois de receber a mensagem de `agreement`; e iv) um nó identifica que o seu número da ordem da configuração global está desatualizado.

Se dois ou mais controladores iniciam o protocolo simultaneamente, apenas um deles é capaz de alcançar adequadamente $n/2 + 1$ mensagens de confirmação (`ok`). No caso de mais de dois iniciadores, pode acontecer de nenhum deles atingir o número necessário de votos e todos recebam uma mensagem de `cancel`, que indica a existência de outra transação em andamento. Nesse caso, cada controlador, que recebe a mensagem `cancel`, relança a sua requisição de atualização após a espera por um tempo aleatório.

Se o controlador iniciador falhar antes de enviar qualquer mensagem `agreement`, não há danos para o estado atual do protocolo, pois não existe qualquer transação iniciada. Por outro lado, se ele falhar depois enviar qualquer mensagem `agreement`, um controlador que recebeu a mensagem, a reenvia, acrescentando a sua própria assinatura na mensagem, e a marca como uma mensagem de recuperação. O novo controlador iniciador envia a mensagem `agreement` reassinada a um novo grupo de $n/2 + 1$ controladores. No caso de a atualização de política já haver sido instalada na rede, o novo iniciador apenas atualiza seu banco de dados de políticas, instala a política nos comutadores que controla e propaga a informação. Caso contrário, ele segue o protocolo como se fosse o iniciador da requisição.

O caso de uma falha de um controlador é tratado de duas formas diferentes. Primeiro, se um controlador falha após receber a mensagem de `agreement`, o controla-

dor iniciador aguarda sua resposta até que um tempo limite ou, se a falha interrompe a conexão TCP, o iniciador detecta imediatamente a perda de conexão. Em seguida, o iniciador envia uma nova mensagem `agreement` para outro controlador até que alcança $n/2 + 1$ votos positivos, mensagens de `ok`, ou pelo menos uma mensagem `cancel`. O segundo caso é quando o controlador não participa em qualquer acordo entre os controladores. Nesse caso, a falha é ignorada. Quando um controlador recupera-se de uma falha, ele atualiza seu banco de dados de políticas com qualquer outro controlador ativo na rede.

Se todos controladores seguem o protocolo sem falhas, esse caso não é viável. Contudo, um controlador pode falhar e não atualizar sua base de dados de políticas. Um controlador sabe que está desatualizado quando recebe uma mensagem com o número de ordem de política maior do que a que ele está esperando. A mensagem pode ser `agreement`, `cancel` ou `commit`. Quando recebe uma `agreement` ou `commit`, o controlador desatualizado pede ao controlador atualizado, que a enviou a mensagem, a base de dados mais recente das políticas da rede. No caso de uma mensagem de `cancel`, o controlador desatualizado aborta a etapa de efetivação da política e, em seguida, atualiza sua base de dados com o controlador que o enviou a mensagem com o número mais recente de ordem de políticas.

Vale ressaltar que o protocolo proposto é mais simples do que outros protocolos de consenso, como Paxos [Prisco et al. 2000] e Zab [Junqueira et al. 2011], já que descarta a etapa de votação de líderes e flexibiliza as restrições de durabilidade. A proposta assume alguns detalhes de implementação, como o uso de conexões TCP que mantém o estado da conexão de cada controlador e assegura a confiabilidade e a ordenação das mensagens. Assim, a proposta também garante uma maior disponibilidade do que o protocolo de efetivação de transação de duas fases (*Two-Phase Commit*), apesar de ainda alcançar a efetivação da transação em dois tempos de ida e volta (RTT). Destaca-se ainda que a proposta não assume qualquer mecanismo de sincronização ou qualquer interface de consenso entre os controladores. A solução de compromisso assumido nesta proposta é aumentar a complexidade em relação ao número de etiquetas de marcação de versões de configuração da rede em relação à simplicidade do consenso e ao desprezo ao monitoramento de quais marcações continuam válidas ou não na rede. Considera-se que os nós não possuem um limite superior sobre o número de ordem de política, apesar de considerá-lo um contador cíclico.

O algoritmo de composição de políticas. O algoritmo proposto funciona localmente como mostrado no Algoritmo 1. A ideia principal é de buscar qualquer tipo de conflito (total, parcial ou conflito de atualização) que possa aparecer entre a nova requisição de atualização de política π e o conjunto de políticas já definidas na rede Π, exemplificado na Figura 2(c). Dessa forma, assim que a requisição de atualização de política chega ao controlador, o algoritmo verifica localmente se o domínio da requisição, $dom(\pi)$, está em conflito com a união do domínio de todas as outras políticas já instaladas, $\cup\, dom(\pi_i), \forall \pi_i \in \Pi$. Se o algoritmo identifica um conflito, parcial ou total, o controlador recusa a requisição de atualização de política. Assim, se a requisição chegou ao controlador através de uma mensagem `agreement`, o controlador a recusa através de resposta com a mensagem `cancel`, sinalizando a existência de um conflito. Se a nova política a ser instalada é verificada e conclui-se que é livre de conflito, Expressão 2, ou é

uma atualização explícita, o algoritmo aceita a nova política e, então, a política é instalada, sem qualquer modificação em relação à sua proposição inicial. Vale ressaltar, que a proposta instala as políticas na rede sob uma abordagem de tudo-ou-nada (*all-or-nothing*), em que a política ou é totalmente aceita ou totalmente recusada. Não há a aceitação parcial de políticas. Esse comportamento é desejado, pois garante que não há a possibilidade de geração de estados intermediários e inconsistentes.

Algoritmo 1: Algoritmo de Composição de Atualização de Políticas. O algoritmo executa localmente em cada controlador. A saída do algoritmo é o voto do controlador, a favor ou contra, a instalação da política verificada.

Entradas: π_i (requisição de atualização de política)
Π (conjunto de todas as políticas já instaladas)
conflito := Falso
for $\pi_j \in \Pi$ **do**
  if $dom(\pi_i) \cap dom(\pi_j) \neq \emptyset$ ***e*** $isUpdate(\pi_i, \pi_j) = False$ **then**
    conflito := Verdadeiro
  end
end
Saída : conflito

O Algoritmo 1 verifica se os domínios de atuação das políticas se sobrepõem. Contudo, o algoritmo pode ser aplicado, sem perda de generalidade ou corretude, com a adoção de métodos mais complexos de identificação de conflitos e composição de regras de forma mais elaborada de políticas, como através do uso a linguagem `Pyretic` [Monsanto et al. 2013].

Prova de corretude. Para provar a corretude de funcionamento da proposta, é necessário provar que a instalação de políticas na rede respeitam duas propriedades: i) as políticas são serializadas e ii) as políticas instaladas não estão em conflito com qualquer outra política na rede. Assim, primeiro prova-se por contradição que a ordem global de instalação das políticas, definida entre os controladores, é a mesma que a ordem local de qualquer controlador na rede. Após, prova-se usando o mecanismo de indução que a composição de todas as políticas é consistente.

Teorema 1: A ordem local de instalação de políticas em qualquer controlador da rede é compatível com a ordem global.

Prova por contradição. Assume-se que a ordem local de instalação de políticas em um dos controladores da rede não é compatível com a ordem global de instalação. Assim, assume-se que existe o controlador c_i, em que a política π_2 precede a política π_1, $\pi_2 <_{Hc_i} \pi_1$, e, na ordem global, π_1 precede π_2, $\pi_1 <_H \pi_2$. Como qualquer outro controlador, diferente de c_i, é compatível com a ordem global, tem-se que existe um controlador c_j, em que a ordem local é $\pi_1 <_{Hc_j} \pi_2$. As c_i e c_j executam corretamente o protocolo proposto e não é possível de ocorrer reordenamento de mensagens na rede. Para instalar a política π_2 antes da instalação de π_1, o controlador c_i teve que obter no mínimo $n/2 + 1$ votos de outros controladores (mensagens `ok`), assim como os demais controladores tiveram que obter $n/2 + 1$ votos para instalar π_1 antes de π_2. De acordo com o protocolo, um controlador

não pode votar em ordens contraditórias. Assim, o único modo possível de haver duas ordens locais diferentes é através da votação das ordens por dois conjuntos disjuntos de controladores votantes. Nesse caso, seriam necessários $(n/2+1)+(n/2+1) = n+2$ votos. Como a rede apresenta apenas n controladores, esse é um cenário impossível e, então, prova-se que uma ordem local diferente da ordem global é uma contradição lógica.

Teorema 2: A composição das políticas é livre de conflitos.

Prova por indução. O mecanismo de indução é usado sobre o conjunto Π, que representa o conjunto de todas as políticas instaladas na rede.

`Caso base` ($dom(\Pi_0) = \emptyset$) **:** Nesse caso, o conjunto Π é vazio e, então, é trivialmente um conjunto de políticas não-conflitantes.

`Hipótese indutiva` ($dom(\Pi_i) = \cup_{k=1}^{i} dom(\pi_k)$) **:** Seja Π_i o conjunto de todas as políticas instaladas até a requisição π_i, para $i > 0$. Assim, o domínio de Π_i é definido como a união dos domínios de todas as políticas no conjunto. Por hipótese, considera-se que a composição de todas as políticas em Π_i é consistente.

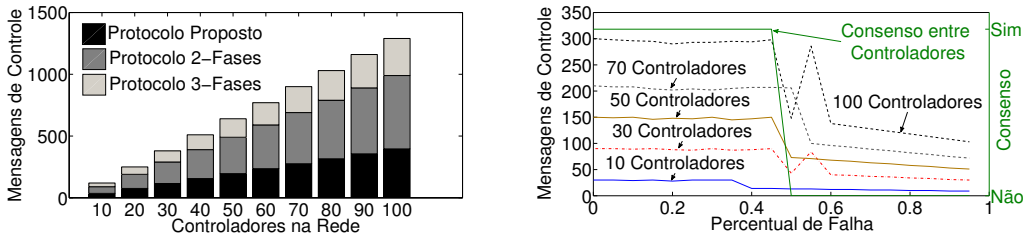
`Passo indutivo` ($dom(\Pi_{i+1}) = dom(\Pi_i) \cup dom(\pi_{i+1})$) **:** Considera-se que todas as políticas em Π_i são compostas de maneira consistente, como é previsto na hipótese indutiva. Assim, a prova consiste em mostrar que a inclusão da política π_{i+1} em Π_i não gera conflitos. Para tanto, usa-se o algoritmo proposto de verificação de conflitos. O algoritmo executa localmente e a resposta do algoritmo é se o controlador deve votar a favor ou contra a aceitação da requisição de atualização de política. Adicionar π_{i+1} ao conjunto Π_i só é possível se, e somente se, $n/2+1$ controladores votarem a favor, garantindo que não há conflitos entre a nova política π_{i+1} e todas as demais políticas em Π_i. De acordo com o Teorema 1, no mínimo um controlador, entre os $n/2+1$ controladores que votam, deve já ter instalado todas as políticas em Π_i e, portanto, está de acordo com a ordem global de políticas. Se todos os $n/2+1$ controladores aprovam a nova política significa que a nova política π_{i+1} não entra em conflito com nenhuma outra política instalada na rede, já que ao menos um controlador no grupo de controladores votantes conhece todas as políticas já aprovadas. Caso contrário, a política π_{i+1} é completamente rejeitada. Logo, prova-se o Teorema 2, mostrando que a composição de Π_i com π_{i+1} só é possível se não houver conflitos.

5. Resultados Experimentais

O protocolo de consistência proposto foi avaliado através da simulação do consenso entre os nós controladores de uma SDN com controle distribuído, aplicando-se uma extensão para o cenário distribuído do simulador SDN desenvolvido por Mattos *et al.* [Mattos et al. 2016b, Mattos e Duarte 2015].

Um protótipo do mecanismo foi implementado para avaliar a carga de mensagens trocada entre os nós. Nesse experimento, consideram-se, a critério de comparação, os protocolos de efetivação de transações de duas fases (*Two Phase Commit* – 2PC) e de três fases (*Three Phase Commit* – 3PC). A Figura 3(a) compara o número de mensagens enviadas pelos protocolos de efetivação de duas fases (`2PC`), efetivação de três fases (`3PC`) e o protocolo de consenso proposto (`Proposto`) para a instalação de uma política na rede. As topologias consideradas são malhas completas de 10 a 100 nós controladores. Os resultados evidenciam que a quantidade de mensagens enviadas pelo protocolo pro-

posto é menor que a enviada pelo protocolo de efetivação de duas fases. Ao se considerar 30 controladores, por exemplo, a redução no número de mensagens de controle chega a 25%, quando comparado com o protocolo de efetivação de duas fases e 50%, com o de três fases. Considerando n o número de nós que estão executando os protocolos, a análise do comportamento de cada protocolo revela que o número esperado de mensagens, em um cenário sem falhas, para o `2PC` é de $4 \times (n-1)$ mensagens e para o `3PC`, $6 \times (n-1)$ mensagens. O protocolo proposto, por sua vez, apresenta no máximo $3n$ mensagens.



(a) Mensagens para uma atualização. (b) Mensagens enviadas em cenários de falhas.

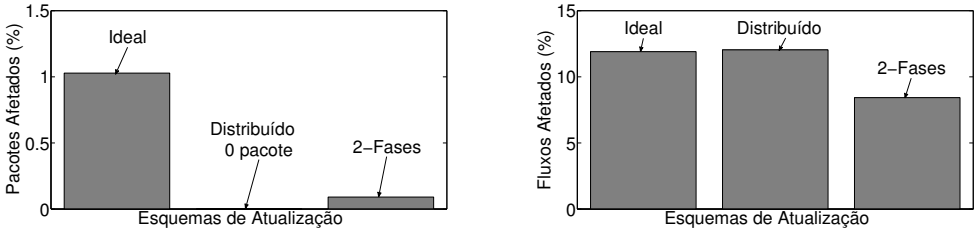
Figura 3. Comparação da carga de mensagens de controle gerada pelo protocolo proposto (`Proposto`) com os protocolos de efetivação de duas fases (`2PC`) e três fases (`3PC`). a) Mensagens de controle em função do número de controladores, a proposta reduz de 25% em relação `2PC` para um número de controladores maior que 30. b) Mensagens de controle em função do percentual de falha de nós da rede. A proposta mantém um baixo número de mensagens e converge mesmo até quando quase metade da rede falha. `Sim` indica o consenso pela aceitação da atualização e `Não`, o consenso pelo aborto da operação.

O experimento seguinte avalia a resiliência do protocolo proposto à ocorrência de falhas na rede. A Figura 3(b) mostra o número de mensagens enviado na rede quando há falhas nos nós. Os nós alcançam o consenso, mesmo quando o índice de falhas na rede é próximo a 50%, $n/2 - 1$ nós falham. Caso a maioria dos nós falhem, as requisições não são aceitas e, então, são abortadas pelo protocolo. A Figura 3(b) evidencia a baixa carga de mensagens na rede, mesmo quando as transações são abortadas. No caso em que a proposta aborta a efetivação das políticas na rede, o número de mensagens é reduzido e apresenta pouco impacto no funcionamento normal da rede. Quando ocorrem falhas, o protocolo proposto envia novas mensagens a nós aleatórios até exaurir a busca por nós ativos ou conseguir o número necessário de votos. Contudo, essa busca pode gerar a expiração do tempo limite de espera por uma reposta dos controladores ativos e que já responderam. Por essa razão, verifica-se a incidências de picos de envio de mensagens nos cenários em que as falhas na rede estão próximas a 50% dos nós controladores, Figura 3(b).

Na segunda etapa de avaliação da proposta, foi simulada uma SDN, baseada na topologia real de rede da RNP, no Brasil, com 31 nós[4]. Os parâmetros da simulação definem que a chegada de novos fluxos é uniformemente distribuída entre todos os nós da rede e o intervalo de chegada entre fluxos segue uma distribuição *log-normal* com média 7 ($\mu = 7$) e desvio padrão igual a 2 ($\sigma = 2$) [Mattos et al. 2016b, Mattos e Duarte 2015]. A chegada de fluxos acontece durante os 900 primeiros passos de simulação, cada fluxo

[4]O grafo da topologia da rede foi obtido em *The Internet Topology Zoo*, disponível em http://www.topology-zoo.org/.

é modelado com uma duração de 50 passos e o fim da simulação é determinado quando não há mais pacotes ou eventos a serem tratados.



(a) Percentual de pacotes em trânsito que são encaminhados por duas configurações distintas.

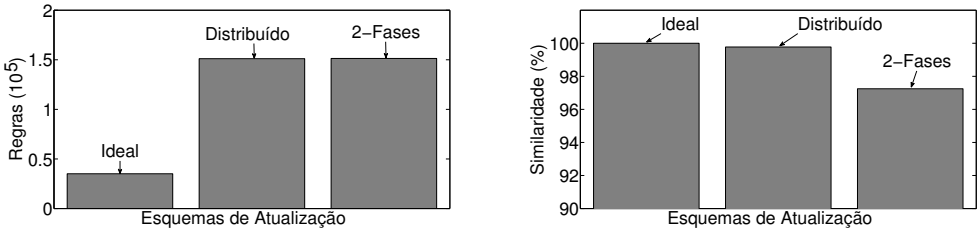
(b) Percentual de pacotes que são encaminhados por duas configurações distintas.

Figura 4. Impacto das atualizações nos pacotes e fluxos encaminhados na rede.

A simulação do esquema distribuído foi realizada definindo-se um controlador para cada nó da rede e todos os controladores executam o protocolo de consenso. O protocolo de consenso acorda quanto à versão da configuração da rede e a distribui entre os demais controladores. O esquema de atualização com controle distribuído baseado no protocolo de consenso (`Distribuído`) foi comparado com os esquemas centralizados de atualização de duas fases (`2-Fases`) e atualização ideal (`Ideal`). O ideal é factível somente em um cenário simulado, pois considera que todo o encaminhamento de pacote é interrompido durante o processo de atualização das regras de encaminhamento nos comutadores. Contudo, a atualização ideal é considerada como um esquema de atualização proporcional, ou seja, aquele em que o custo de instalação da atualização é proporcional às mudanças implementadas [Reitblatt et al. 2012]. Assim, ao comparar um esquema de atualização com a atualização ideal, verifica-se o quão próximo o esquema proposto está de uma atualização proporcional.

A Figura 4(a) compara o comportamento dos esquemas de atualização de política em relação aos pacotes encaminhados na rede. O esquema distribuído não encaminha nenhum pacote por mais de uma versão de configuração da rede, comportamento esperado para um esquema consistente por pacote. Contudo, durante a atualização de duas fases, verifica-se a ocorrência de um pequeno percentual de pacotes que é encaminhado por mais de uma configuração de rede. Isso ocorre porque o modelo de controlador considerado é o mais ingênuo possível, em que após a atualização ele sempre marca os pacotes com a nova configuração de rede sem guardar qualquer estado. Assim, pacotes de fluxos que ainda não foram instalados em comutadores intermediários, podem chegar a um controlador já atualizado, a partir de então, são encaminhados por uma nova configuração [Mattos et al. 2016b]. A atualização por controladores distribuídos age mais prontamente na rede do que a atualização de duas fases com controle centralizado, apresentando um resultado mais próximo ao ideal. O efeito é evidenciado pelo número de fluxos afetado pelas atualizações na rede, mostrado na Figura 4(b). O esquema distribuído atualiza 42% mais de fluxos do que o centralizado de atualização de duas fases.

O número total de regras instaladas nos comutadores da rede é evidenciado na Figura 5(a). Essa métrica indica o quanto da memória dos comutadores é usada por cada esquema de atualização. Como os esquemas de atualização distribuído e de atualização de duas fases centralizado instalam regras no núcleo da rede para garantir a consistência



(a) Número total de regras instaladas na rede. (b) Similaridade com o ideal.

Figura 5. Comparação do número de regras e do efeito no destino causado pelo uso dos esquemas de atualização. a) Número total de regras instaladas na rede. b) A similaridade das propostas em relação ao esquema ideal.

por pacote, o número de regras instalado por esses esquemas chega a ser 4x superior ao do ideal[5]. Por sua vez, a Figura 5(b) compara o resultado do encaminhamento no destino dos pacotes. A similaridade mede o quão próximo o encaminhamento dos pacotes em cada esquema de atualização está do ideal. Essa medida fornece uma estimativa da qualidade de cada esquema de atualização. Verifica-se que o esquema de duas fases já apresenta um desempenho muito próximo do ideal. Contudo, a proposta do esquema distribuído alcança um resultado ainda mais próximo do ideal devido à coordenação eficiente de ações entre os controladores com o uso do protocolo de consistência proposto.

6. Conclusão

Atualizações de políticas de forma consistente em uma Rede Definida por Software com plano de controle distribuído é um desafio, pois as requisições devem ser ordenadas globalmente e as políticas, compostas sem conflitos. Esse artigo propõe um protocolo de consistência para controladores distribuídos, em que o conflito entre políticas é verificado localmente e a ordem global de instalação é garantida através do acordo entre controladores. O artigo propõe ainda um algoritmo simples para a composição de políticas que se aproveita da interface de consenso provida pelo protocolo de consenso. O algoritmo é executado localmente e sua interação com o protocolo de consistência proposto assegura que todas as políticas aceitas são livres de conflitos. A prova de corretude do protocolo e do algoritmo propostos é realizada através de verificação formal. A simulação de um cenário de aplicação da proposta em uma topologia de rede real mostra que o número de mensagens do protocolo proposto é inferior ao das demais propostas, mesmo em cenários de falha, e que a proposta alcança atualizações consistentes em dois tempos de ida e volta. Os resultados mostram ainda que a proposta alcança o consenso e, consequentemente, atualizações consistentes sem a ocorrência de impasses, mesmo quando até $n/2 - 1$ controladores apresentam falhas. A simulação da aplicação do protocolo proposto em uma topologia de rede real mostra que o esquema distribuído de atualização de políticas aumenta em até 42% o número de fluxos que são tratados pela configuração mais recente da rede e mantém a garantia de que cada pacote em trânsito é consistentemente encaminhado por apenas uma configuração de rede.

[5]Há fluxos não expiraram na tabela de fluxos dos comutadores e são afetados por mais de uma atualização, gerando um aumento ainda maior no número de regras instaladas do que a instalação de uma regra a mais por fluxo em cada comutador.

Referências

[Canini et al. 2015] Canini, M., Kuznetsov, P., Levin, D., Schmid, S. et al. (2015). A distributed and robust SDN control plane for transactional network updates. Em *The IEEE INFOCOM 2015*.

[Ferguson et al. 2012] Ferguson, A. D., Guha, A., Liang, C., Fonseca, R. e Krishnamurthi, S. (2012). Hierarchical policies for software defined networks. Em *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN'12, páginas 37–42, New York, NY, USA. ACM.

[Gray e Lamport 2006] Gray, J. e Lamport, L. (2006). Consensus on transaction commit. *ACM Trans. Database Syst.*, 31(1):133–160.

[Han et al. 2015] Han, J. H., Mundkur, P., Rotsos, C., Antichi, G., Dave, N., Moore, A. e Neumann, P. (2015). Blueswitch: enabling provably consistent configuration of network switches. Em *Architectures for Networking and Communications Systems (ANCS), 2015 ACM/IEEE Symposium on*, páginas 17–27.

[Junqueira et al. 2011] Junqueira, F. P., Reed, B. C. e Serafini, M. (2011). Zab: High-performance broadcast for primary-backup systems. Em *IEEE/IFIP 41st International Conference on Dependable Systems Networks (DSN), 2011*, páginas 245–256.

[Katta et al. 2013] Katta, N. P., Rexford, J. e Walker, D. (2013). Incremental consistent updates. Em *ACM SIGCOMM - HotSDN'13*, Hong Kong, China. ACM.

[Levin et al. 2012] Levin, D., Wundsam, A., Heller, B., Handigol, N. e Feldmann, A. (2012). Logically centralized?: state distribution trade-offs in software defined networks. Em *Proceedings of the First workshop on Hot topics in software defined networks*, HotSDN'12, Helsinki, Finland. ACM.

[Luo et al. 2015] Luo, S., Yu, H. e Li, L. (2015). Consistency is not easy: How to use two-phase update for wildcard rules? *Communications Letters, IEEE*, 19(3):347–350.

[Mattos e Duarte 2015] Mattos, D. M. F. e Duarte, O. C. M. B. (2015). Atualizaçao reversa: Garantindo consistência de estados em redes definidas por software. Em *SBSeg 2015 - XV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, Florianópolis - Brazil.

[Mattos et al. 2016a] Mattos, D. M. F., Duarte, O. C. M. B. e Pujolle, G. (2016a). A resilient distributed controller for software defined networking. Em *IEEE ICC 2016 - Next Generation Networking and Internet Symposium (ICC'16 - NGN)*, Kuala Lumpur, Malaysia.

[Mattos et al. 2016b] Mattos, D. M. F., Duarte, O. C. M. B. e Pujolle, G. (2016b). Reverse update: A consistent policy update scheme for software-defined networking. *IEEE Communications Letters*, 20(5):886–889.

[McClurg et al. 2015] McClurg, J., Hojjat, H., Cerny, P. e Foster, N. (2015). Efficient synthesis of network updates. Em *ACM SIGPLAN - PLDI*, Portland, USA. ACM.

[McGeer 2012] McGeer, R. (2012). A safe, efficient update protocol for openflow networks. Em *ACM SIGCOMM - HotSDN'12*, Helsinki, Finland. ACM.

[Monsanto et al. 2013] Monsanto, C., Reich, J., Foster, N., Rexford, J., Walker, D. et al. (2013). Composing software defined networks. Em *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI'13)*, páginas 1–13, Berkeley, CA, USA. USENIX Association.

[Prisco et al. 2000] Prisco, R. D., Lampson, B. e Lynch, N. (2000). Revisiting the Paxos algorithm. *Theoretical Computer Science*, 243(1-2):35 – 91.

[Reitblatt et al. 2012] Reitblatt, M., Foster, N., Rexford, J., Schlesinger, C. e Walker, D. (2012). Abstractions for network update. Em *Proceedings of the ACM SIGCOMM 2012*, páginas 323–334, New York, USA. ACM.

Mobility-Flow: Solução para Handover Transparente e com Suporte à Autenticação 802.1x em Redes OpenFlow

Edivaldo C. de A. Junior, Edson A. M. Avelar, Kelvin L. Dias, Paulo R. F. Cunha

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
CEP: 50740-540 – Recife – PE – Brasil

{ecaj,eama,kld,prfc}@cin.ufpe.br

Abstract. *User authentication is an essential element to ensure adequate levels of security in accessing the strategic corporate services. However, this procedure can lead to the degradation of the quality perceived by the user upon handover executions between access points since reauthentication is required. Despite that numerous mobility management strategies exist in the literature, user authentication is neglected and it is not taken into account as part of the solution design. This article proposes an OpenFlow-based mobility management solution for Wi-Fi networks, considering 802.1x authentication. The results from testbed experiments show the benefits of the proposed solution to best effort and video traffics.*

Resumo. *Autenticação do usuário é um elemento primordial para garantir níveis adequados de segurança no acesso aos serviços corporativos estratégicos. Contudo, este procedimento pode acarretar degradação da qualidade percebida pelo usuário quando das execuções de handover entre pontos de acesso, uma vez que a reautenticação é requerida. Apesar de inúmeras soluções presentes na literatura para o gerenciamento de mobilidade, a autenticação do usuário é negligenciada. Este artigo propõe uma solução de gerenciamento mobilidade baseada no OpenFlow para redes Wi-Fi, considerando autenticação 802.1x. Os resultados mostram os benefícios da solução proposta para tráfego de melhor esforço e de vídeo.*

1. Introdução

Com a crescente utilização de tecnologias de redes sem fio e o aumento no acesso à Internet vias *smartphones*, *tablets* e *notebooks* em qualquer lugar e a qualquer momento, bem como a convergência de várias tecnologias para redes IP, cada vez mais, o usuário faz uso de dispositivos móveis para dispor de serviços e aplicações em seu dia-a-dia.

A ampliação dos pontos de acesso de rede sem fio gera novas questões e desafios quanto ao gerenciamento eficiente e centralizado, bem como ao provimento de vários requisitos atuais, tais como: continuidade do serviço enquanto o usuário se desloca entre pontos de acesso, procedimento denominado de *handover*, e também aspectos de Qualidade de Serviço/Experiência (QoS – *Quality of Service*/QoE – *Quality of Experience*), aliado à necessidade do cumprimento de requisitos de segurança, como

a autenticação do usuário, um elemento primordial para garantir níveis adequados de segurança no acesso aos serviços estratégicos da organização. Apesar das inúmeras soluções existentes na literatura para o gerenciamento de mobilidade e *handover* (Ferretti, et al., 2016), o suporte à autenticação do usuário é um aspecto negligenciado nesses cenários, fazendo com que requisitos de QoS e QoE não possam ser garantidos devido aos atrasos inerentes ao processo de reautenticação quando o dispositivo migra para um novo ponto de acesso ou rede.

Com o advento do paradigma SDN (*Software-Defined Networking*) e com penetração cada vez maior no mercado de equipamentos com a tecnologia OpenFlow(OF), tornou-se possível prover soluções inovadoras. SDN baseia-se no princípio da separação entre os planos de controle e dados e esse controle é totalmente programável (McKeown et al., 2008). Nesta arquitetura temos a vantagem da simplificação dos ativos de rede, pois o plano de controle deste é transferido para o controlador SDN, no qual são executadas as aplicações que podem atribuir aos ativos, funcionalidades de roteadores e/ou protocolos diversos para o funcionamento de rede. Estas aplicações podem ser desenvolvidas em uma linguagem de propósito geral, viabilizando a inovação e soluções antes apenas permitidas e implementadas pelos próprios fabricantes.

Nesse contexto, soluções baseadas no protocolo IP, que não foram amplamente difundidas pelos fabricantes que utilizam suas próprias soluções fechadas, podem ser redesenhadas considerando os benefícios do emprego do paradigma SDN. Assim, a gerência de mobilidade pode utilizar-se desta filosofia de rede aberta e programável, para viabilizar a implementação de ideias promovidas pela abordagem baseada em NetLMM (*Network-based Localized Mobility Management*) (Internet Engineering Task Force, 2010), tais como PMIP (*Proxy Mobile IP*), onde a gerência de mobilidade é realizada com a isenção de sinalização nos dispositivos clientes, ficando esta sinalização a cargo do núcleo da rede.

Este artigo propõe, implementa e avalia o desempenho de uma estratégia de *handover* que reduz a latência do processo de reautenticação utilizando técnica de transferência de contexto de informações de segurança, bem como, considera uma arquitetura SDN para programar dinamicamente o tratamento da mobilidade com suporte dos ativos pertencentes ao núcleo da rede, o que viabiliza a continuidade do serviço e requisitos de QoS/QoE das aplicações, além de evitar o envolvimento do dispositivo móvel na sinalização para troca de ponto de acesso.

Este artigo está organizado da seguinte forma. Na Seção 2, os trabalhos relacionados serão discutidos. Em seguida, na Seção 3, a arquitetura da proposta, denominada *Mobility-Flow*, será detalhada. Na Seção 4, será apresentada a avaliação da proposta e a Seção 5 mostra as considerações finais deste trabalho.

2. Trabalhos Relacionados

Esta Seção aborda trabalhos relacionados no que concerne o gerenciamento de mobilidade baseado em redes definidas por software.

Devido à demanda crescente de gerenciamento de mobilidade em redes sem fio, o paradigma SDN tem recebido cada vez mais a atenção também nessa área. A plataforma OpenRoads (Yap *et al.*, 2009) foi a primeira abordagem criada para

manipular redes sem fio definidas por software no padrão OpenFlow. Foi projetada para fornecer suporte às novas abordagens de gerenciamento de mobilidade que antes eram difíceis de testar em ambientes de produção.

A abordagem CloudMAC (Dely et al., 2012) foi criada para gerenciar redes públicas do padrão IEEE 802.11, sem perder a capacidade de ser extensível, permitindo que novos serviços possam ser implementados facilmente em linguagens de programação de alto nível. No CloudMAC, os pontos de acesso físicos apenas encaminham pacotes até os dispositivos móveis. Processamento e gerenciamento dos pacotes são realizados em pontos de acesso virtuais localizados em nuvem computacional. As ligações entre os pontos de acesso físicos e virtuais são gerenciadas através de um controlador OpenFlow, que redireciona a transmissão dos fluxos. A desvantagem dessa abordagem é que o handover pode ser prejudicado devido ao alto atraso decorrente do processamento dos pacotes nos pontos de acesso virtuais nos datacenters.

O trabalho proposto em (Avelar et al., 2013) propõe o PMIPFlow, solução para o gerenciamento de mobilidade baseada em SDN. Um mecanismo de antecipação de handover baseado em lógica fuzzy para a redução das quedas de conexões durante o handover é elaborado e avaliado. Apesar de sua contribuição, o PMIPFlow como outras iniciativas, também não considera aspectos de segurança e suporte à autenticação. Além disso, os pontos de acesso precisam embarcar parte do protocolo para desempenhar as funções específicas para o provimento da mobilidade. A proposta utiliza uma versão de *software switch* que é executado no espaço do usuário, o que contribui para um baixo desempenho em relação à vazão máxima nominal dos pontos de acesso utilizados.

Em (Tantayakul et al., 2016), os autores defendem a não utilização de uma implementação baseada em protocolos legados como o PMIP e sim uma estratégia puramente baseada em SDN (SDN *Mobility*), que pode utilizar uma de suas características, a visão global da rede, para tratar e encaminhar fluxos de forma a entregar o serviço de mobilidade. Para avaliação da proposta foi criado um cenário simulado no mininet e [1]realizado comparativo entre PMIPv6 padrão e a solução proposta, demonstrando ao fim que a solução SDN *Mobility* é mais vantajosa em termo de perdas de pacotes.

O OpenSWDN (J. S. Zender et al., 2015), introduz uma arquitetura Wi-Fi baseada em uma abordagem SDN/NFV, e permite o controle programável e a virtualização de recursos físicos do WiFi através do uso do LVAP (light virtual access point). A estratégia adotada tem como desvantagem o fato do handover ser prejudicado caso haja alto atraso decorrente do processamento dos pacotes nos pontos de acesso virtuais nos datacenters. No Mobility-Flow apenas as primeiras mensagens são enviadas ao controlador, que instala regras de fluxo, responsáveis pelo encaminhamento do trafego.

Em suma, os trabalhos relacionados apresentados não consideram o processo de autenticação em suas propostas ou o fazem com grande custo computacional. A maioria das propostas baseiam-se em ambiente emulado, modelagem analítica ou solução com baixo desempenho usando OpenFlow como aplicação no espaço do usuário. A

[1] Mininet é um emulador de redes - links http://www.mininet.org/

abordagem de virtualização dos roteadores sem fio na nuvem ainda requer mecanismos visando garantir a QoS para aplicações de tempo real. Além disso, o trabalhos não exploram avaliações de métricas de QoE.

3. Arquitetura proposta

A Figura 1 apresenta a proposta de arquitetura para gerenciamento de mobilidade.

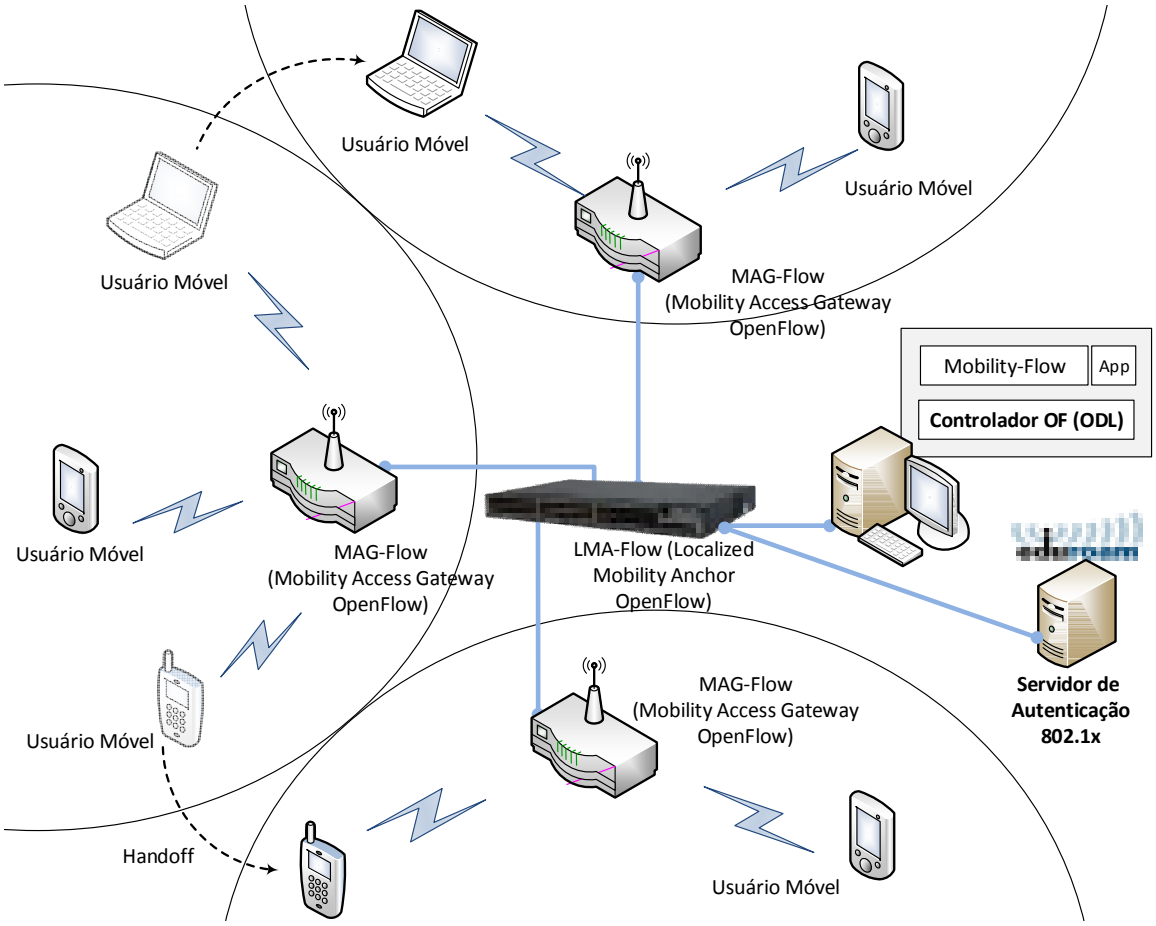


Figura 1. Arquitetura Mobility-Flow

O MAG-Flow (*Mobility Access Gateway - OpenFlow)* é o elemento responsável por monitorar e gerenciar a mobilidade dos usuários e possui duas representações: física e lógica. A parte física do MAG-Flow corresponde aos roteadores comerciais modificados. Nesses roteadores, o *firmware* original é substituído pelo OpenWRT,[2] que é um sistema linux para dispositivos com limitações de recursos computacionais. Além do firmware, é acrescentada uma versão OpenFlow instalada no espaço do *kernel*. Foi escolhida a versão 1.3 do OpenFlow, pois as versões anteriores não dão suporte a funcionalidades de QoS, que poderão ser utilizadas em trabalhos futuros da proposta.

Os elementos denominados de LMA-Flow (*Localized Mobility Anchor* – OpenFlow) são gateways dos elementos MAG-Flow, cuja responsabilidade é gerenciar

[2] Distribuição GNU/Linux para pontos de acesso sem fio - https://openwrt.org/

todo o tráfego, além de manter estruturas de dados que permitem saber se o usuário está se movendo entre MAG-Flows distintos, isto é, executando *handover*.

3.1 Sinalização de Conexão

A sinalização de conexão, mostrada na Figura 2, possui quatro etapas: autenticação, mobilidade, fornecimento de IP e implementação de regras OpenFlow, como detalhado nas trocas de mensagens:

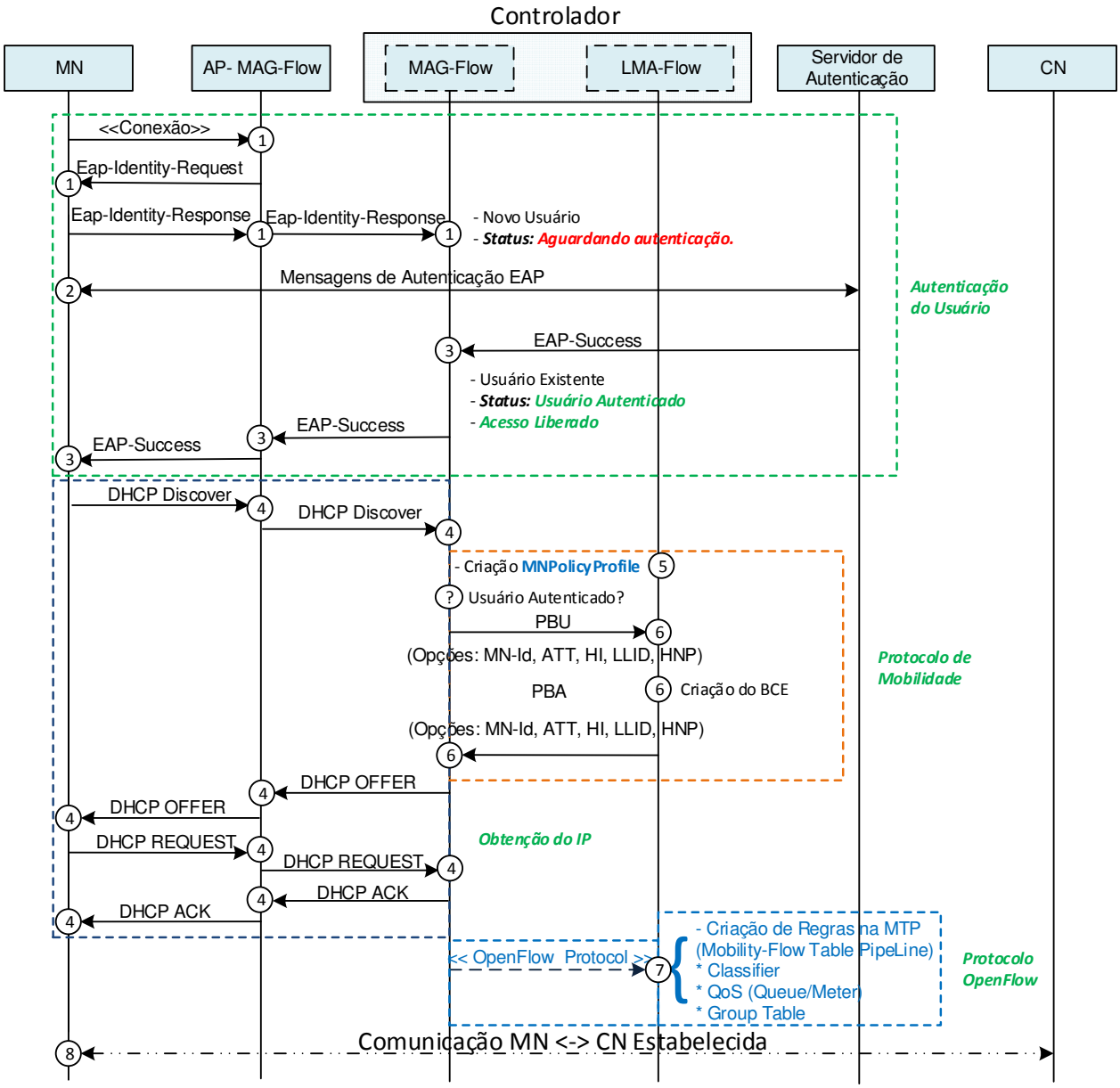


Figura 2. Sinalização de Conexão

[1]: Solicitação de Conexão a Rede Sem Fio: O MN (*Mobile Node*) conecta-se à rede autenticada, fornecendo nome de usuário e senha. No protótipo, a rede autenticada usa o protocolo EAP (*Extensible Authentication Protocol*) com autenticação TTLS (*Tunneled Transport Layer Security*). A comunicação entre o

cliente e o MAG-Flow é feita através do protocolo EAP e o protocolo RADIUS é usado para a troca entre o roteador e o servidor de autenticação.

[2]: Monitoramento de mensagens de Autenticação: A aplicação MAG-Flow executada no Controlador SDN, monitora toda a troca de mensagens entre cliente e servidor de autenticação, até a confirmação da autenticação do usuário.

[3]: Obtenção de *status* da autenticação: O servidor de Autenticação informa à aplicação MAG-Flow residente no controlador o sucesso na autenticação do usuário e este informa para o MN.

[4]: Processo de obtenção de IP: Nessa etapa o MN troca mensagens com o servidor DHCP implementado no controlador, para obtenção de endereço IP. Esta comunicação é gerenciada pelo protocolo de mobilidade do MAG-Flow.

[5]: Criação da Estrutura de dados de autenticação e controle de acesso: Nesta etapa é criada a estrutura de dados *MNPolicyProfile* no controlador, com informações de autenticação e permissões de acesso do MN, para serem utilizadas pelos MAG-Flow e LMA-Flow para autorização e controle dos MNs durante o Handover.

[6]: Troca de Mensagens do protocolo de Mobilidade: As mensagens PBU (*Proxy Binding Update)* e PBA (*Proxy Binding Ack*) são trocadas entre MAG-Flow e LMA-Flow com objetivo de verificar permissões e possíveis ocorrências de *handover* e criação e atualização do BCE.

[7]: Implementação de Regras OpenFlow: Nessa etapa, as regras OpenFlow para roteamento e encaminhamento são implementadas nos switches para provimento da comunicação.

[8]: Estabelecimento da comunicação entre MN e CN: Por fim, a comunicação entre o nó móvel (MN) e o nó correspondente (CN) é estabelecida.

3.2 Sinalização de *Handover*

A sinalização de handover, destacada na Figura 3, é semelhante à de conexão. A diferença é que quando o MN tentar migrar do MAG-Flow1 para o MAG-Flow2, o protocolo de mobilidade verifica que o usuário requisitante já estava conectado à outra rede. Então o MAG-Flow2 consulta o LMA-Flow para verificar se a solicitação trata-se de uma *handover*. Em caso positivo, o LMA-Flow verifica na estrutura de armazenamento MNPolicyProfile, os identificadores utilizados na primeira conexão do MN solicitante, e autoriza a migração do cliente sem a necessidade de uma reautenticação, pois a identificação é realizada a partir dos identificadores repassados por meio da transferência destes, dentro do mesmo contexto de segurança, conforme preceitua a RFC3374 (*Context Transfer Problem Statement - Internet Engineering Task Force*, 2002), o que corrobora para um *handover* suave, sem quebras decorrentes do tempo gasto no processo de reautenticação.

Durante o handover, o endereço obtido pelo usuário é o mesmo utilizado na rede anterior. Isso faz com que não haja quebra de conexão devido à mudança de endereçamento do MN na nova rede, desse modo, mantendo a conexão nas camadas superiores.

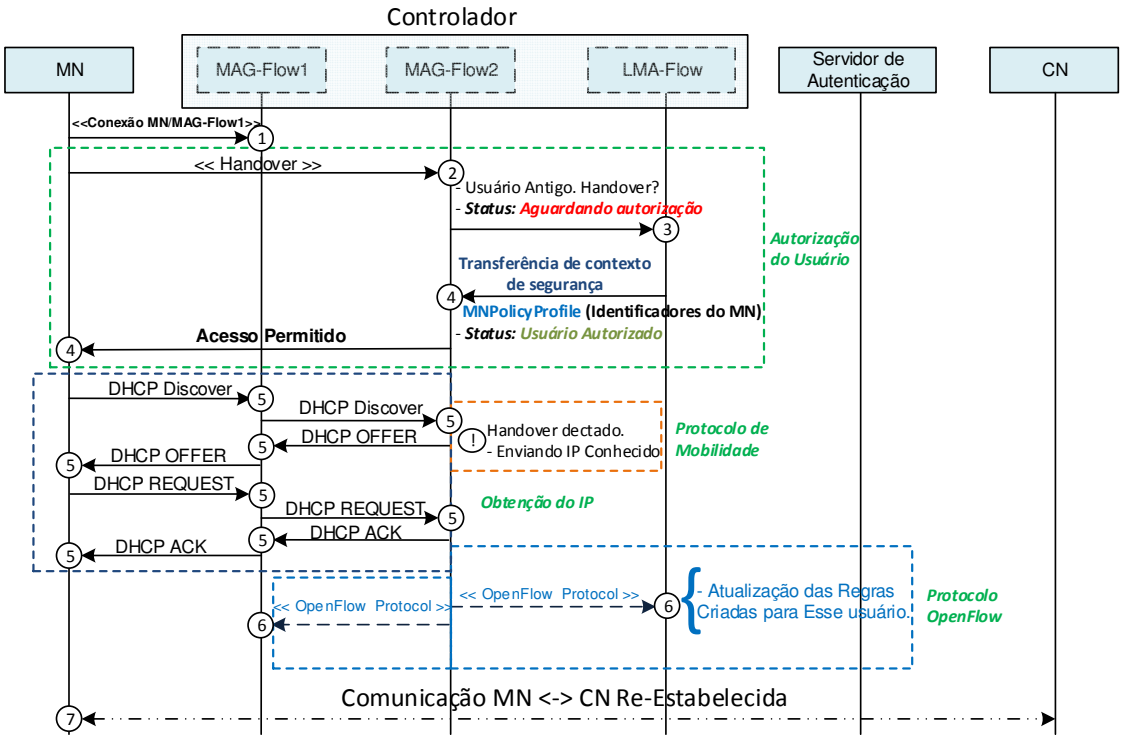


Figura 3. Sinalização de Handover

Todo processo de handover é detalhado nas trocas de mensagens abaixo:

[1]: Estado inicial do MN: O nó móvel (MN) inicialmente está conectado no MAG-Flow1, e segue em deslocamento aproximando-se do próximo ponto de acesso sem fio, o MAG-Flow2.

[2]: Solicitação de troca de ponto de acesso (*Handover*) devido à mobilidade: O nó móvel(MN) conectado no MAG-Flow1, solicita conexão para MAG-Flow2.

[3]: Verificação de nova conexão ou Handover: Ao receber a solicitação de conexão, o MAG-Flow2 envia uma mensagem ao LMA-Flow, para consultar se a solicitação trata-se de uma Handover.

[4]: Transferência de contexto de segurança: O LMA-Flow verifica que o MN estava conectado no MAG-Flow1 e que a solicitação de conexão para o MAG-Flow2 não refere-se a uma nova conexão e sim um handover, assim este consulta a estrutura de dados MNPolicyProfile para verificação das informações de autenticação e permissões de acesso do MN e informa ao MAG-Flow2 sobre a autorização do MN, sem a necessidade da realização de novo processo de autenticação. Por fim, o MAG-Flow2 de posse da autorização repassada pelo LMA-Flow, aceita a solicitação do MN.

[5]: Processo de obtenção de IP: Nessa etapa o MN troca mensagens com o servidor DHCP implementado no controlador, para obtenção de endereço IP. Esta comunicação é gerenciada pelo protocolo de mobilidade do MAG-Flow. Durante o processo de *handover,* o endereço obtido pelo MN é o mesmo utilizado na rede anterior. Isso faz com que a quebra de conexão da camada 3 decorrente da mudança de endereçamento não ocorra, mantendo a conexão nas camadas superiores.

[6]: Implementação e atualização de Regras OpenFlow: Nessa etapa, as regras de roteamento e encaminhamento correspondentes ao conexão MN/MAG-Flow1 são

retiradas e são implementadas regras OpenFlow nos switches para provimento da comunicação do MN/MAG-Flow2.

[7]: Estabelecimento da comunicação entre MN e CN: Por fim a comunicação entre o nó móvel(MN) e o nó correspondente(CN) é estabelecida.

4. Avaliação da Proposta

Nesta seção é realizada a avaliação da proposta em um testbed 802.11ac. Na subseção 4.1 é apresentado todo ambiente de teste e as funções desempenhadas pelos equipamentos durante os experimentos. Em seguida, na subseção 4.2, são apontados os resultados obtidos nos experimentos com a utilização da estratégia de gerenciamento de mobilidade com transferência de contexto de segurança implementada na proposta.

4.1 Ambiente de Teste

O testbed possui três pontos de acesso *Mobility-flow*, que são roteadores sem fio com OpenvSwitch (Openvswitch, 2016) instalado para suportar OpenFlow no nível do kernel. O ambiente possui também um comutador *Mobility-Flow*, que possui a função do LMA (*Localized Mobility Anchor*) do protocolo PMIP, ou seja, funciona como ponto de ancoragem e saída dos APs. O comutador Mobility-Flow é um roteador modificado para, assim como os Mobility-Flow-APs, suportar o protocolo OpenFlow. O Controlador é o elemento mais importante da rede e gerencia toda a rede OpenFlow. O último elemento da rede é o Gateway, responsável por redirecionar o tráfego da rede OpenFlow para a Internet. Nos experimentos, o gateway também funciona como servidor de autenticação. A Figura 4 mostra a planta baixa do ambiente onde o testbed foi implantado.

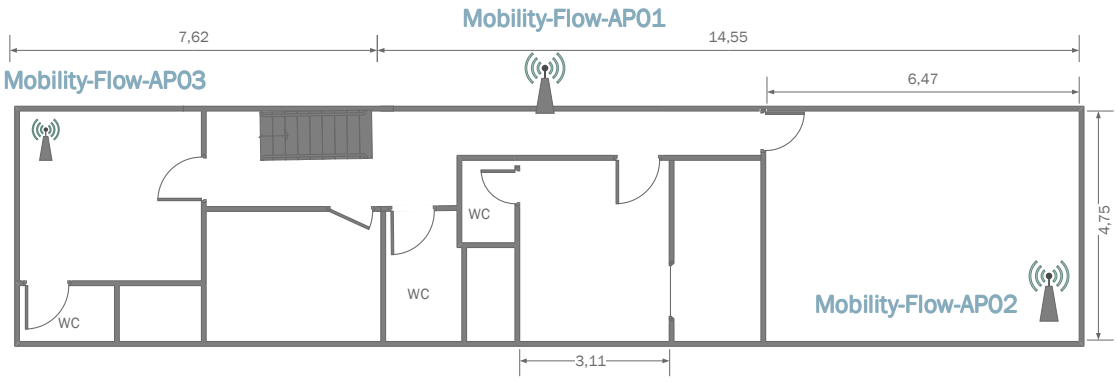


Figura 4. Ambiente do Testbed

Todos os testes foram realizados com um usuário movendo-se a uma velocidade média de um metro por segundo (1m/s). A Figura 5 mostra o padrão de movimentação usado em todos os testes. Escolheu-se esse padrão para comparar de forma justa as diferentes soluções apresentadas. Um percurso completo da Figura 5 leva em média 120 segundos. Por isso, todos os testes foram ajustados para durarem 200 segundos.

Para cada teste foram realizadas 30 repetições, como forma de obter resultados com relevância estatística.

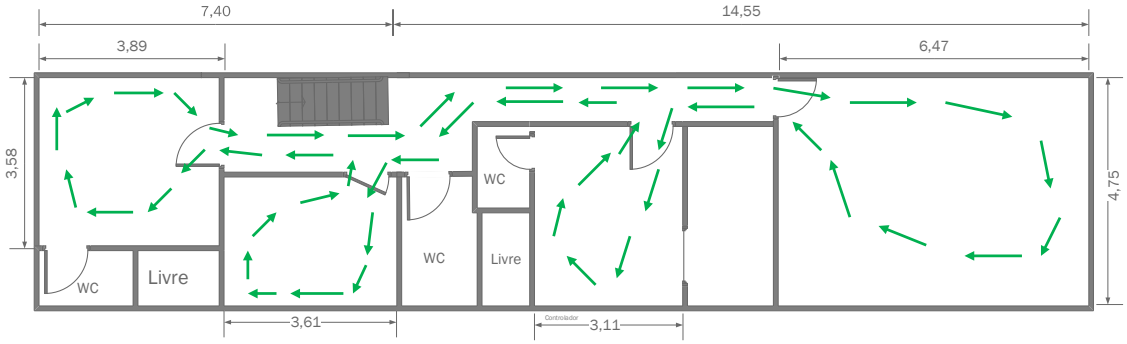


Figura 5. Padrão de movimentação pelo Ambiente

A Figura 6 mostra o mapa de força de sinal dos três pontos de acesso sem fio. O mapa foi feito com auxílio da ferramenta Heat mapper[3] da Ekahau, e mostra como está a distribuição do sinal no testbed. Quanto mais próximo do verde maior a intensidade do sinal, quanto mais próximo do vermelho, menor a intensidade. As potências das antenas dos pontos de acesso foram levemente reduzidas para permitir o cenário mostrado na Figura 6, caso contrário, devido ao espaço limitado, o usuário poderia se manter conectado mesmo estando no cômodo mais afastado. Como mostrado na Figura 6, quando o usuário migra de um cômodo ao outro ele é forçado a mudar de rede, do contrário a conexão é perdida.

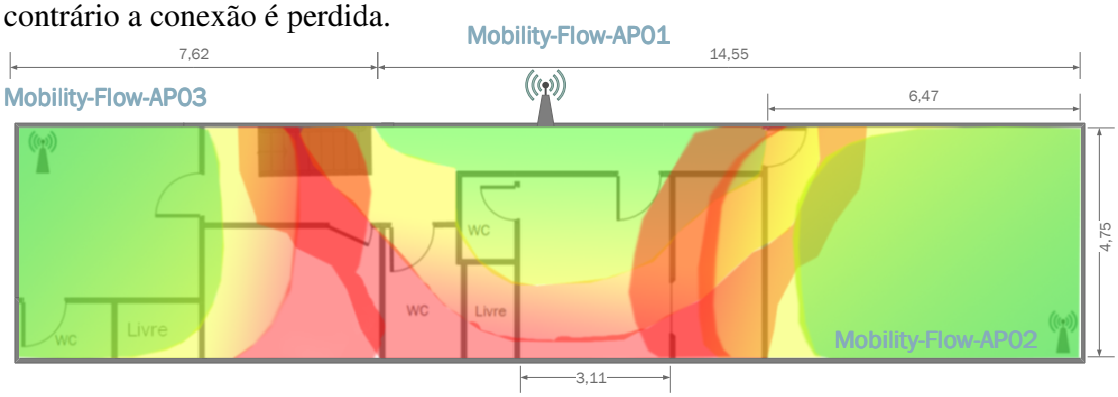


Figura 6. Mapa de Força do Sinal do Testbed

4.2 Resultados

4.2.1 Avaliação de QoS: Vazão e Atraso médio

Para a obtenção dos resultados das avaliações de vazão e atraso médio, foram realizados dois experimentos:

- **Vazão:** O experimento consistiu na execução de um fluxo sintético UDP de 100 Mbps entre o gateway e o cliente em deslocamento e realização da coleta dos resultados das vazões alcançadas. A Figura 7 mostra a média da vazão do switch do OpenvSwitch nos experimentos, onde, sem a proposta de transferência de contexto, a média da vazão foi de 64,24Mbps e com a proposta o valor médio da vazão aumenta para 74,43 Mbps, equivalente a um ganho de 15,8% em relação

[3] http://www.ekahau.com/wifidesign/ekahau-heatmapper

aos resultados alcançados na avaliação sem a adição da proposta de transferência de contexto de segurança.

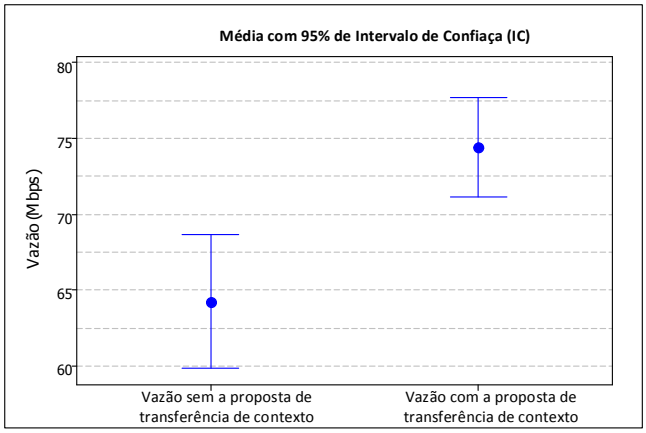


Figura 7. Vazão UDP com e sem transferência de contexto de segurança

- **Atraso Médio:** Para o segundo experimento foi enviado um *streaming* de vídeo do gateway para o cliente em deslocamento, para verificação do atraso entre os frames do vídeo. Quanto menor o atraso, melhor é a qualidade do vídeo recebido. A Figura 8 compara o atraso médio em 30 repetições, onde verifica-se que, sem a utilização da proposta de transferência de contexto de segurança, as perdas de pacotes devido à mobilidade do usuário são mais acentuadas, com isso o atraso médio da entrega dos pacotes fica em torno de 40ms. Já com a utilização da proposta de transferência de contexto de segurança, o valor de atraso médio cai para 30ms, o que representa um ganho de 25% nos resultados que utilizam a proposta. Os resultados de atraso médio apresentados na Figura 8 foram calculados usando a ferramenta Evalvid (Evalvid, 2016).

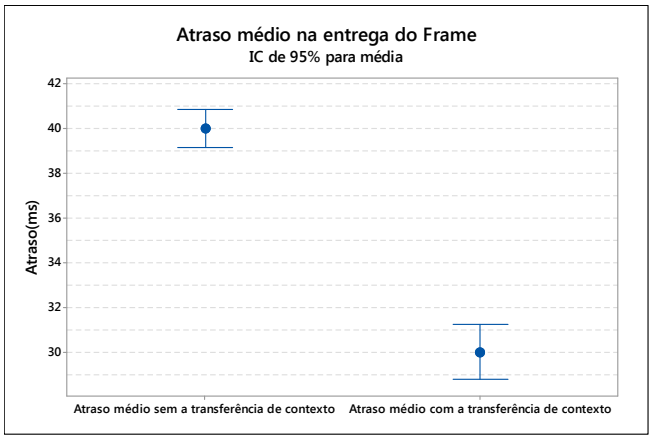


Figura 8. Atraso médio na entrega do Frame

4.2.2 Avaliação de QoE: PSNR

O PSNR (*Peak Signal-to-Noise Ratio*) é uma métrica de QoE que estima a qualidade do vídeo em decibéis, comparando o vídeo original com o vídeo recebido pelo usuário. Para cada faixa de valores de PSNR, há uma qualificação para o vídeo que foi recebido pelo usuário.

Para este experimento os vídeos foram enviados um de cada vez. Durante a transmissão do vídeo, assim como na avaliação de QoS, o usuário permanece em constante movimento de um ponto de acesso para outro. Após o recebimento do vídeo foi utilizada a versão gratuita da ferramenta MSU (*Video Quality Measurement Tool*), (MSU, 2013), para avaliar e extrair as informações para avaliação do vídeo recebido.

A Figura 9 mostra que o valor médio do PSNR sem a proposta de transferência de contexto de segurança foi de 26,7, considerado ACEITÁVEL, já com a adição da transferência de contexto de segurança o resultado de PSNR foi de 32,2, considerado BOM. Na comparação entre os intervalos de cada cenário, com 95% de confiança. É possível notar que com a proposta o PSNR é visivelmente superior ao cenário sem a proposta de transferência de contexto.

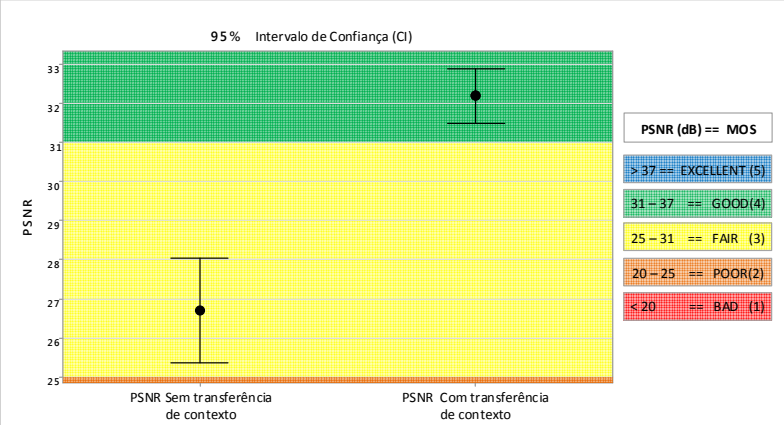


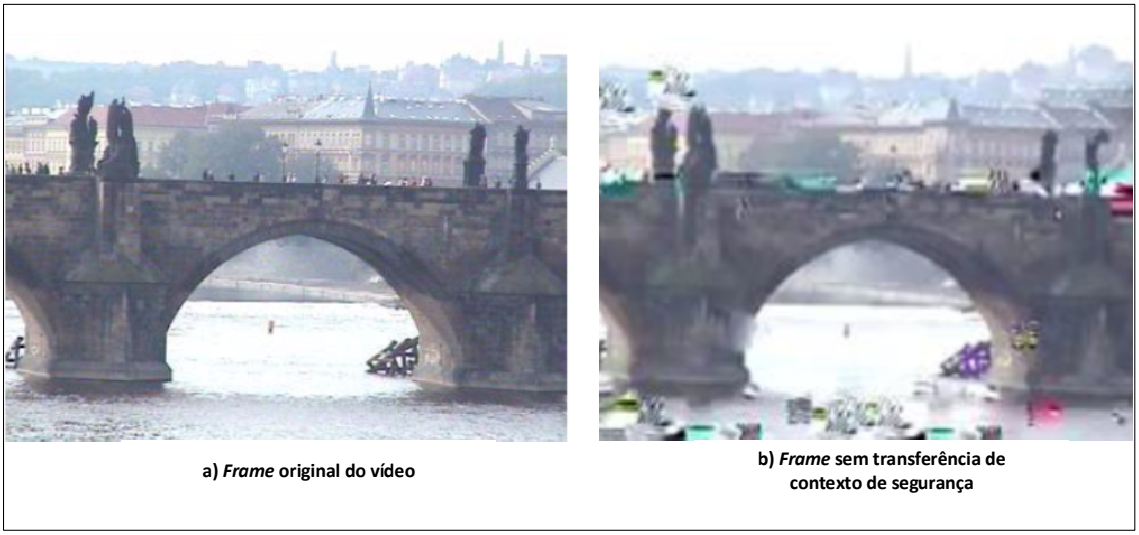
Figura 9. Atraso médio na entrega do Frame

a) Respaldo Visual PSNR

A Figura 10 apresenta um *frame* original do vídeo usado na avaliação do PSNR. O vídeo utilizado dura 30 segundos e 2000 frames. Esse vídeo foi escolhido por durar mais do que os outros vídeos disponíveis em (Bridge, 2016). Como um percurso completo dura 120 segundos em média (ver Figura 5), o mesmo vídeo foi repetido 5 vezes, para gerar uma duração de 150 segundos.

Figura 10. *Frame* original do vídeo

A Figura 11 apresenta uma comparação para respaldo visual do *frame* original do vídeo (a), *frame* no momento do handover sem a proposta de transferência de contexto de segurança (b). É possível observar que o quadro mostrado na Figura 11b está com uma qualidade bem inferior, se comparado com o *frame* original mostrado na Figura 11a. Isso ocorre porque houve perdas de pacotes durante o handover, que foram acentuadas pelo processo de reautenticação, o que culminou na reconstrução incompleta do *frame* no receptor.

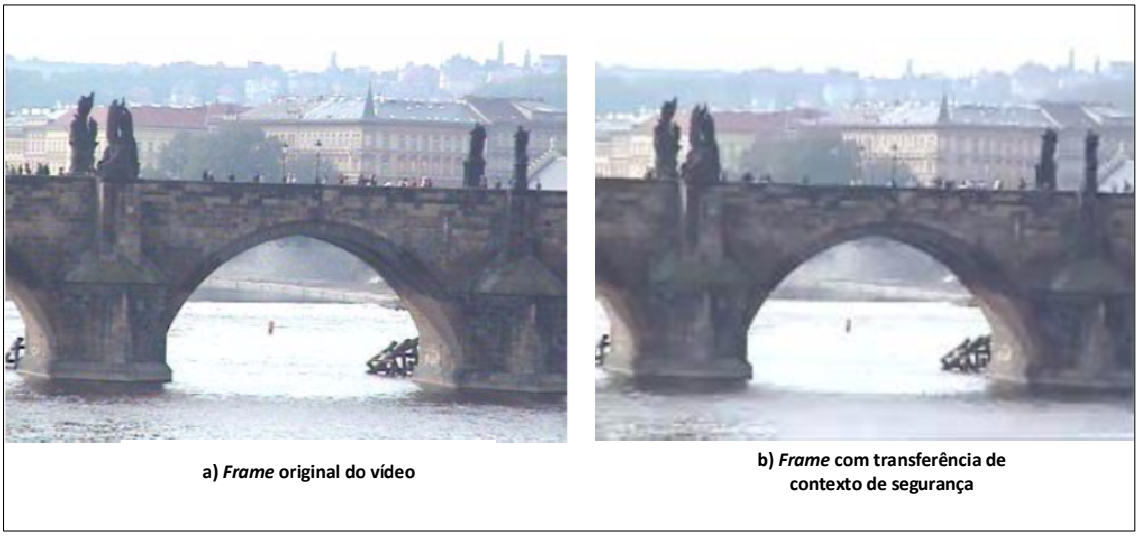


a) *Frame* original do vídeo

b) *Frame* sem transferência de contexto de segurança

Figura 11. *Frame* original (a) e *frame* sem a transferência de contexto (b)

Por outro lado, a Figura 12, é uma comparação para respaldo visual do *frame* original do vídeo (a), e o *frame* capturado no momento do handover com a proposta de transferência de contexto de segurança (b). Percebe-se que nesse caso, o *frame* capturado está com qualidade melhor que na Figura 11b, pois poucos pacotes são perdidos com a proposta, o que resulta em um *frame* de vídeo mais completo. Mesmo com perdas da camada L2, o vídeo resultante do cenário com a proposta é muito superior ao vídeo sem a proposta.



a) *Frame* original do vídeo

b) *Frame* com transferência de contexto de segurança

Figura 12. *Frame* original (a) e *frame* com transferência de contexto (b)

5. Conclusão

Este trabalho implementou e avaliou uma proposta de estratégia de transferência de contexto de segurança para o *Mobility-Flow*, que suaviza a degradação do processo de reautenticação decorrente do handover entre redes e assim fornece níveis aceitáveis de QoE. Foram discutidas diversas iniciativas na literatura voltadas a prover o serviço mobilidade, com destaque para a lacuna no aspecto referente ao suporte à segurança nas soluções de *handover*.

A avaliação foi realizada em um ambiente de experimentação e a proposta obteve como resultados os seguintes ganhos: 15,8% na vazão, 25% no atraso médio e 20,5% no PSNR em relação ao cenário de não utilização da proposta de transferência de contexto de segurança. Os resultados obtidos demonstram a aplicabilidade da proposta no gerenciamento mobilidade seguro, bem como sua eficácia no suporte aos requisitos de QoS/QoE para sessões de tráfego de vídeo de usuários móveis.

6. Referências

Avelar, E. A. M., Marques, L., Dias, K. L.. PMIPFlow: *Uma Proposta para Gerenciamento de Mobilidade em Redes Definidas por Software*. In: WPerformance, 2013, Maceió. WPerformance, 2013. p. 1-14.

Bridge (close) (2016). http://www2.tkn.tu-berlin.de/research/evalvid/cif.html

Dely, P., Vestin, J., Kassler, A., Bayer, N., Einsiedler, H., and Peylo, C. (2012). *"CloudMAC – An OpenFlow based Architecture for 802.11 MAC Layer Processing in the Cloud."* In Globecom Workshops (GC Wkshps), 2012 IEEE, pages 186–191. IEEE.

Evalvid 2016. A Video Quality Evaluation Tool-set, http://www2.tkn.tu-berlin.de/research/evalvid/EvalVid/docevalvid.html

Ferretti, S., Ghini, V. and Panzieri, F.. "A survey on handover management in mobility architectures." Computer Networks 94 (2016): 390-413.

Internet Engineering Task Force (2010). Network-based Localized Mobility Management. http://datatracker.ietf.org/doc/charter-ietf-netlmm/

Internet Engineering Task Force (2002). Context Transfer Problem Statement. https://www.ietf.org/rfc/rfc3374.txt

J. S. Zander, C. Mayer, B. Ciobotaru, S. Schmid, A. Feldmann, OpenSDWN: programmatic control over home and enterprise WiFi, in: Proceedings of the ACM SIGCOMM SOSR, Santa Clara, CA, USA, 2015.

K. Tantayakul, R. Dhaou and B. Paillassa, "Impact of SDN on Mobility Management," *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, Crans-Montana, 2016, pp. 260-265.

MSU Video Group (2013). Video Quality Measurement Tool. http://compression.ru/index_en.htm.

N. McKeown, T. Anderson, H. Balakrishnan,G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. "*OpenFlow : enabling innovation in campus networks*". ACM SIGCOMM Computer Communication Review. April de 2008, Vol. 38, 2, pp. 69-74.

OpenvSwitch (2016). OpenFlow 1.3 kernel Switch. Disponível em: http://openvswitch.org/download/.

Yap, K.-K., Kobayashi, M., Underhill, D., Seetharaman, S., Kazemian, P., and McKeown, N. (2009). The Stanford OpenRoads Deployment. In Proceedings of the 4th ACM International Workshop on Experimental Evaluation and Characterization, WINTECH '09, pages 59–66, New York, NY, USA. ACM.

Wang, Y., & Bi, J. "*A Solution for IP Mobility Support in Software Defined Networks*." Computer Communication and Networks (ICCCN), 2014 23rd International Conference on, pages 1 – 8.

Comparação de Políticas de Divisão de Tráfego em Data Center empregando SDN

Erik de Britto e Silva[1,2], **Henrique Moura**[1], **Daniel Fernandes Macedo**[1], **Luiz F. M. Vieira**[1], **Marcos A. M. Vieira**[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

{erik,henriquemoura,damacedo,lfvieira,mmvieira}@dcc.ufmg.br

[2]Departamento de Computação e Sistemas
Universidade Federal de Ouro Preto (UFOP) – Campus João Monlevade
João Monlevade, MG – Brasil

erik@decsi.ufop.br

Abstract. *It is estimated that Internet traffic will triple in five years, which will increase server response time. One way to reduce such time is to balance the load on replicated servers. This work compares five load balancing policies using Software Defined Networks (SDN) in an OpenFlow switch: round robin, random, txbytes (transmitted bytes), cpuq-load (CPU usage and number of open connections), and load/load-prev (load forecast with switch statistics). These policies consider limitations such as the cost to retrieve network statistics and to install new rules. The results show that the txbytes and cpuq-load policies outperformed the others. On the other hand, the load-prev policy proved to be promising when adjusted for traffic.*

Resumo. *Estima-se que o tráfego da Internet triplicará em cinco anos, o que aumentará o tempo de resposta dos servidores. Uma forma de reduzir esse tempo é balancear a carga em servidores réplica. Este trabalho compara cinco políticas de balanceamento de carga em Redes Definidas por Software (SDN) em um switch OpenFlow: round robin, random, txbytes (bytes transmitidos), cpuq-load (taxa de CPU e número de conexões abertas) e load/load-prev (previsão com estatísticas do switch). As políticas consideram limitações como o custo para obter estatísticas da rede e instalar novas regras. Os resultados mostram que as políticas txbytes e cpuq-load superaram as demais. Já a política load-prev mostrou-se promissora ao se fazer ajustes em função do tráfego.*

1. Introdução

O volume global de dados trafegados na Internet tem crescido, como mostra o relatório *Cisco Visual Networking Index*[1]. O relatório prevê que o tráfego ultrapassará 1,1 Zettabytes (10^{21} bytes) em 2016, alcançando 2 Zettabytes em 2019. O tráfego global na nuvem (*cloud computing*) se multiplicará cerca de 3 vezes entre 2014 e 2019, passando de 3,4 Zettabytes para 10,4 Zettabytes [2]. Portanto, o tráfego dos data centers é o de maior

[1]http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html

[2]http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.pdf

magnitude entre os citados.

Segundo [Cardellini et al. 2002], o desempenho das aplicações WEB percebido pelos usuários finais está sendo dominado pela latência dos servidores e, ao mesmo tempo, a capacidade da rede está aumentando mais rapidamente do que capacidade dos servidores. Isto indica que o uso combinado de vários servidores respondendo pelo mesmo serviço em um enlace de alta velocidade é adequado para se obter uma solução. Os desafios são, portanto, atender o volume futuro de tráfego nos data centers, bem como o crescente tráfego atual e suas oscilações, através do aumento da velocidade. Apenas aumentar a capacidade dos enlaces físicos para aumentar a taxa de transmissão dos dados, é de alto custo, não é escalável e implica no emprego de servidores mais rápidos, portanto mais caros. Uma alternativa é utilizar vários enlaces paralelos entre origem e destino, cada enlace conectado a um servidor réplica e empregar um método para a divisão do tráfego total entre os enlaces. Estes servidores réplica podem ser servidores de prateleira sem grandes modificações [Al-Fares et al. 2008], portanto de menor valor do que servidores especiais com características de maior desempenho.

Mesmo com mecanismos de otimização do atendimento ao tráfego, data centers são provisionados com capacidade em excesso [Xu et al. 2014] e sem escalabilidade. Fora do horário de pico ocorre o problema da subutilização de recursos, o que causa desperdício [Armbrust et al. 2010]. Empresas menores podem obter soluções escaláveis sem desperdício, como por exemplo o *AWS — Amazon Web Services* da Amazon [Ferraris et al. 2012], que oferece serviços e infraestrutura completa de data center por demanda. Porém, tais soluções podem não atender a todos os perfis de instituições. Portanto, algumas soluções são implementadas através de investimentos em infraestrutura física própria, que pode utilizar as políticas implementadas neste trabalho.

Em redes definidas por software (SDN) [Guedes et al. 2012, Macedo et al. 2015], o plano de controle (onde são tomadas as decisões de encaminhamento) é separado do plano de dados (onde o encaminhamento é feito). O plano de controle é executado em um controlador que pode rodar algoritmos diversos e portanto pode ser utilizado para engenharia de tráfego. Para utilizarmos engenharia de tráfego em SDN, por exemplo, um controlador OpenFlow será o responsável pela seleção dos caminhos [Jain et al. 2013]. O uso do protocolo OpenFlow permite controlar diretamente os fluxos, possibilitando que parâmetros de gerência de tráfego (banda, caminhos, *QoS*, etc.) sejam utilizados. O custo típico de uma solução para divisão de tráfego em um *middlebox* de 20 Gbps está em torno de US$ 80.000,00 [Patel et al. 2013], o dobro do custo de um switch OpenFlow 40 Gbps com um controlador, tomando como base a solução de um switch OpenFlow Xtreme Networks X770 (US$ 38,400,00) e um controlador Dell Core™ i7-6700 com 16 GB de memória (US$ 709,00) - preços da AMAZON[3].

Este trabalho compara diferentes políticas de balanceamento de carga, as quais são baseadas em redes definidas por software e têm como principal motivação o crescente volume de tráfego de entrada em um data center. Tais políticas utilizam Engenharia de Tráfego [Leduc et al. 2006] e visam diminuir o tempo médio de duração das requisições de serviço, dividindo-as entre vários servidores réplica. Dessa forma, mais requisições podem ser atendidas em um mesmo intervalo de tempo. Para evitar a subutilização, po-

[3]http://www.amazon.com, 17/Dez/2016

derão existir implementações de mecanismos para desligamento de servidores e switches – economia de energia e de vida útil. Por ser ambiente SDN, os recursos físicos não utilizados podem ser alocados para outras aplicações ou serviços, virtualizados ou não.

Todas as políticas aqui apresentadas realizam monitoramento de carga, cuja definição varia de uma política para outra. A carga pode ser: (1) volume de dados já trafegados nas portas de um switch, (2) utilização de CPU e o número de conexões abertas em cada servidor, (3) volume de dados já trafegados na interface de rede de cada servidor, (4) volume de dados nas portas de um switch, predito por meio de estatísticas do switch. Para ilustrar os resultados foram escolhidas requisições de serviço HTTP para realizar os experimentos executados.

A contribuição do presente trabalho é a comparação de cinco políticas de balanceamento de carga através da utilização de Engenharia de Tráfego em Redes Definidas por Software em um switch OpenFlow comercial. As políticas consideram limitações como o custo para obter estatísticas da rede e instalar novas regras.

O restante deste artigo está organizado da seguinte forma: Na Seção 2 são apresentados os ambientes de trabalhos relacionados. A Seção 3 apresenta as soluções a serem comparadas. O ambiente experimental e os resultados obtidos são apresentados na Seção 4. Por fim, a Seção 5 conclui este trabalho.

2. Trabalhos Relacionados

De forma similar ao presente trabalho, Akyildiz *et al*. utilizam *Differentiated Services* (DIFFSERV) e MPLS (*Multiprotocol Label Switching*) para implementar *QoS* em um ambiente de testes com TCP/IP [Akyildiz et al. 2003]. Em MPLS, rotas de backup são exigidas, mesmo com Engenharia de Tráfego, para manter *QoS*. A vantagem da Engenharia de Tráfego com SDN é: (1) a agilidade de programar a rede para isolar os enlaces em falha; (2) adicionar escalabilidade e eficiência a *QoS*, pois cada pacote é encaminhado através de um fluxo; (3) não há o custo adicional de reservar banda para cada classe diferente de tráfego como em DIFFSERV; e (4) o desempenho é aumentado, pois, em DIFFSERV, se uma classe tem seu tráfego reduzido ou é removida, não há a liberação da banda para as outras classes de tráfego, em tempo real.

Existem outras soluções similares que utilizam SDN. O ambiente B4, apresentado por Jain *et al*., interliga data centers da Google distribuídos em alguns continentes empregando Engenharia de Tráfego em SDN [Jain et al. 2013]. O B4 visa manter uma alta utilização dos enlaces, priorizar tráfego e prover tolerância a perdas, reescalonando os fluxos perdidos. Neste trabalho a solução apresentada é avaliada em menor escala relativa a quantidade de equipamentos e a complexidade da rede. A solução proposta neste artigo não lida com tolerância a perdas, maiores atrasos ou realocação de banda. Contudo estas características podem ser incorporadas futuramente.

O DUET gerencia data centers que provêm serviços na nuvem, realizando o balanceamento do tráfego utilizando uma solução com switches SDN em software e outros switches não SDN [Gandhi et al. 2014]. O controlador DUET calcula as rotas e distribui a programação das rotas de ECMP (*Equal-Cost Multi-Path*) e de tunelamento para os switches físicos e os de software. DUET fornece dez vezes mais capacidade e dez vezes menos latência quando comparado com soluções inteiramente baseadas em switches de

software para data centers, seu custo é menor e a solução se adapta mais rápido à dinâmica da rede. Ao contrário das soluções apresentadas no presente trabalho, o DUET emprega redundância de switches para alta disponibilidade da rede.

O *"OpenFlow-based Server Load Balancing Gone Wild"* utiliza programação proativa, inserindo regras coringa, de todos os possíveis prefixos dos endereços IPs dos clientes, para os enlaces de um data center [Wang et al. 2011]. Assim, os possíveis fluxos já estão divididos entre os vários caminhos para os servidores réplica. Com isso, poucos pacotes são direcionados ao controlador, resultando em um mínimo impacto na vazão da rede. Tal programação proativa não ocorre nas soluções propostas no presente trabalho.

O *Plug-n-Serve* realiza balanceamento de tráfego dentro de uma rede local composta de switches Openflow implementados em ambiente real [Handigol et al. 2009]. O *Plug-n-Serve* acompanha a localização física e a carga de servidores réplica na rede local, os quais podem ser conectados ou desconectados em diferentes switches ao acaso. Ele também monitora o congestionamento da rede através de medições de latência. Tal solução difere do presente trabalho ao considerar o congestionamento da rede para monitorar a latência e encaminhar os pacotes pelos caminhos com menor tempo de resposta. A latência é um parâmetro que pode ser adicionado futuramente em nossas soluções, como peso em uma política de escolha do melhor caminho até um servidor réplica.

Rodrigues et al. apresentaram o balanceamento de carga SDN a partir da utilização de CPU em cada servidor réplica [Rodrigues et al. 2015]. A solução proposta pelos autores apresenta diversas características semelhantes à solução **cpuq-load** do presente trabalho, com duas características distintas: (1) Rodrigues et al. utiliza em seus experimentos um *Open vSwitch* [Pfaff et al. 2015] em ambiente virtual, ao passo que a solução **cpuq-load** utiliza dispositivos reais; e (2) o presente trabalho compara soluções que utilizam mais parâmetros de entrada para o balanceamento do tráfego, tais como o tráfego em cada porta do switch real, consumo da CPU e número de conexões no servidor réplica, e o número de bytes já transmitidos na interface de rede.

Tabela 1. Comparação de características de ambientes de trabalhos relacionados

	RE	TR	ESC	AD	AP TR	CONG
AMBIENTE DO PRESENTE TRABALHO	X	I	I	+	I	+
Engenharia de Tráfego em Rede Não SDN - DIFFSERV	I	X	X	+	X	X
Engenharia de Tráfego com SDN - B4	X	I	I	I	I	I
Balanceamento de Carga Híbrido com SDN - DUET	X	I	I	I	I	+
OpenFlow Gone Wild	X	I	I	+	X	+
Balanceamento SDN - Plug-n-Serv	X	I	I	+	I	I
Balanceamento de Carga Web em Redes Definidas por Software	X	I	I	+	I	+

LEGENDA = I: Implementada X: Inexistente +: Pode ser adicionada

A Tabela 1 compara as características do ambiente das soluções apresentadas neste trabalho, ambiente de Redes Definidas por Software com Engenharia da Tráfego, com os ambientes dos trabalhos relacionados. Atribui-se o conceito ***RE*** *– Reserva estática de banda* caso a banda disponível seja dividida em faixas estáticas de menores bandas. Já o conceito ***TR*** *– Tempo real* é atribuído se a resposta do sistema às variações é suficientemente rápido para o bom funcionamento do sistema. A solução é considerada ***ESC*** *– Escalável* se a capacidade do sistema varia para atender às flutuações do tráfego. A solução é de ***AD*** *– Alta disponibilidade*, quando o sistema tem mecanismo(s) de resiliência a

falhas. A característica de ***AP TR*** *– Atraso de programação em tempo real*, onde o atraso da aplicação é crítico e pode impactar o funcionamento do sistema. Por fim, solução é ***CONG*** *– Monitora o congestionamento*, se na divisão do tráfego, o congestionamento da rede é considerado. Embora o ambiente apresentado com as soluções do presente trabalho não possuam no momento **RE** e sim uma escalabilidade conforme a carga, esta característica, bem como **AD** e **CONG**, podem ser adicionadas em trabalhos futuros.

3. Soluções de Divisão de Tráfego

Nesta seção são descritas as soluções de divisão de tráfego do presente trabalho.

3.1. Arquitetura

Neste trabalho, o tráfego a ser dividido entre os servidores réplica origina-se em estações clientes que fazem requisições a um serviço em um endereço IP virtual, seguindo a estrutura mostrada na Figura 1. A solução proposta pode ser aplicada para qualquer serviço TCP ou UDP, contudo neste artigo focamos no serviço HTTP. O modo utilizado para encaminhar cada fluxo aos servidores réplica é similar a um sistema Web baseado em *cluster* [Cardellini et al. 2002], que encaminha os fluxos das requisições HTTP de cada cliente, reescrevendo o endereço IP virtual para o endereço IP físico interno de um servidor réplica através dos switches. O respectivo fluxo da resposta do servidor para o cliente também tem o seu endereço IP físico interno reescrito para o endereço IP virtual nos switches da rede ao ser encaminhado para o cliente. Os switches suportam o protocolo OpenFlow, permitindo ao controlador SDN executar uma política ativa de balanceamento de carga. Este controlador determina os caminhos e faz a alocação de fluxos aos mesmos.

A inserção de uma regra no switch OpenFlow ocorre quando surge o primeiro pacote de um novo fluxo de dados em uma porta do switch. Após calcular por quais enlaces o pacote trafegará, o controlador insere no switch a regra de ida deste primeiro pacote, que o encaminha para a porta calculada. Todas as requisições dos clientes são destinadas ao IP virtual que identifica um servidor HTTP virtual. Assim, a cada novo fluxo a política de engenharia de tráfego define o servidor réplica de destino e insere no switch uma regra que reescreve o endereço IP físico do servidor destino no lugar do endereço IP virtual para este fluxo. Para melhor entendimento, são exibidas abaixo duas sequências de fluxo com reescrita de endereço IP:

- Fluxo de tráfego com as requisições dos clientes
  Clientes → IP Virtual → Switch reescreve IP → IP do servidor réplica
- Fluxo de tráfego com as respostas dos servidores réplicas
  Servidor réplica → Switch reescreve IP para IP Virtual → Clientes

3.2. Políticas de Balanceamento de Carga

O balanceamento de carga pode ser executado como um sistema de controle aberto ou fechado. Utilizamos neste trabalho duas políticas clássicas de balanceamento que funcionam como um sistema aberto. Elas são **round robin** e **random**. A política **round robin** alterna sequencialmente os fluxos entre os servidores disponíveis. A política **random** divide aleatoriamente os fluxos entre os servidores disponíveis. Estas políticas, por serem muito simples, foram utilizadas como *baseline* para nosso trabalho. O balanceamento de carga pode ser modelado, também, como um sistema de controle fechado, que

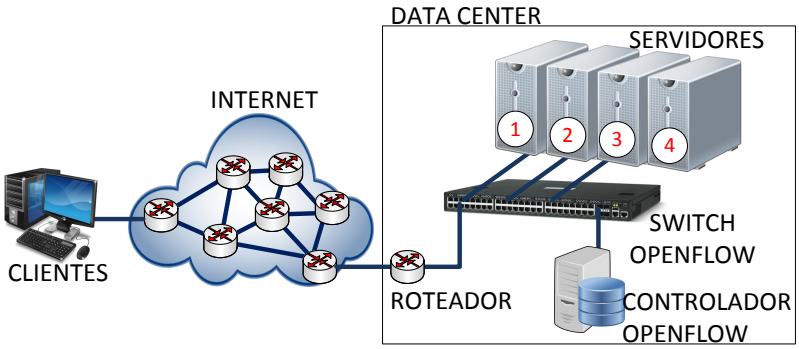


Figura 1. Solução com switch e controlador OpenFlow

utiliza uma variável de controle para atuar sobre o sistema. Neste trabalho apresentamos quatro políticas que utilizam controle de sistema fechado, que são descritas a seguir.

A política **txbytes** utiliza o volume de tráfego (bytes transmitidos) medido na interface da placa de rede de cada servidor. Ela seleciona o servidor com menor quantidade de bytes transmitidos, como mostrado no Algoritmo 1. Em caso de empate, é selecionado o servidor de menor IP. Para obter esta informação foi criada uma rede adicional e independente, conectando o controlador a cada servidor web utilizando um switch tradicional. Uma aplicação cliente-servidor em *sockets* foi implementada para obter os valores da interface do servidor desejado. O tráfego atual obtido sofre um atraso para ser lido, que dependerá da atividade do switch. Para diminuir este problema, pode ser utilizado um esquema de previsão de tráfego futuro, como em **load** e **load-prev-**δ.

Algoritmo 1 Algoritmo implementando a política **txbytes**

1: **function** TXBYTES($replicas$)
2: ▷ $replicas$: conjunto de servidores réplica físicos ordenados pelo IP
3: $load[i] \leftarrow sockcall\_get\_txbytes(replicas[i]), \forall i \in [1, |replicas|]$
4: **return** $\arg\min_i (load[i])$ ▷ servidor com menor carga

A política **cpuq-load** busca informações dos servidores da mesma maneira que **txbytes**, contudo são utilizados três parâmetros para seleção do servidor: a carga da CPU do servidor, o número de conexões na porta do serviço HTTP e o IP do servidor. O pseudocódigo é mostrado em Algoritmo 2. Os parâmetros são avaliados nesta ordem, isto é, selecionam-se os servidores com a menor carga de CPU, como mostrado na linha 4. $servers$ é uma lista dos servidores com carga de CPU mínima ordenada pelo IP. No caso da lista possuir mais de um elemento, é selecionado o servidor com menor quantidade de conexões e, por fim, o de menor número IP. Esta etapa é mostrada na linha 5. Assim, **cpuq-load** sofre o mesmo problema de atraso que **txbytes**.

A política **load** utiliza o volume de tráfego obtido pelo controlador mediante leitura regular das estatísticas do switch via primitiva *Portstats*. *Portstats* é uma chamada do tipo requisição-resposta assíncrona, definida pelo protocolo OpenFlow. O controlador envia uma mensagem solicitando informações sobre o tráfego nas portas do switch OpenFlow e este responde, posteriormente, ao controlador. A chegada da mensagem *Portstats* de resposta no controlador ativa uma função. Nossa aplicação executa a função apresentada em Algoritmo 3. A carga média, utilizada nas políticas **load** e **load-prev-**δ, é

Algoritmo 2 Algoritmo implementando a política **cpuq-load**

1: **function** CPUQ-LOAD($replicas$)
2: ▷ $replicas$: conjunto de servidores réplica físicos ordenados pelo IP
3: $load[i] \leftarrow sockcall\_get\_cpu\_connections(replicas[i]), \forall i \in [1, |replicas|]$
4: $servers \leftarrow \left\{ j \mid load[j].cpu = \min_{i=1}^{|replicas|} load[j].cpu \wedge j \in \{1, |replicas|\} \right\}$
5: **return** $\arg\min_i \ (load[i].num\_connections)$

calculada utilizando uma média móvel ponderada (*EWMA – Exponentially Weighted Moving Average*). A utilização desta média visa evitar a alta taxa de ocupação do controlador ao realizar várias leituras consecutivas em intervalos de tempos curtos, com o objetivo de manter o valor da carga instantânea, e também minimizar a disparidade dos valores utilizados como a carga no controlador entre duas leitura efetivas da carga real no switch [Guo et al. 2014]. Assim, a carga média atual é calculada na linha 10 do Algoritmo 3, onde $nova\_carga_i$ é o valor da carga obtida no instante i (linha 8), o valor da carga média anterior é $carga_{i-1}$ e $\alpha \in [0, 1]$ é o valor de ponderação. A *EWMA* utiliza, portanto, o peso α para ponderar amostras em ordem geometricamente decrescente. Neste trabalho, o valor da carga média é calculado a intervalos regulares de medição, quando a medição das estatístcas de *PortStats* (algoritmo 3) é executado. Portanto, durante o período entre o recebimento das estatísticas por *PortStats*, o valor da carga para cada servidor réplica registrado no controlador permanece constante. A cada novo fluxo, o controlador executa Algoritmo 4.

Algoritmo 3 Algoritmo de *PortStats*

1: **function** _HANDLE_PORTSTATS($\alpha, stats, t, replicas, cargas, tx$)
2: ▷ $stats$: estatísticas retornadas pelo switch ▷ α: fator de ponderação do EWMA
3: ▷ tx: lista com última leitura de bytes transmitidos ▷ t: período entre $PortStats$
4: ▷ $cargas$: carga média calculada anteriormente em $PortStats$
5: ▷ $replicas$: conjunto de servidores réplica físicos ordenados pelo IP
6: **for** $i \leftarrow 1, |replicas|$ **do**
7: $port \leftarrow replica[i].port$
8: $nova\_carga \leftarrow (stats.tx\_bytes[port] - tx[port])/t$
9: $tx[port] \leftarrow stats.tx\_bytes[port]$
10: $cargas[port] \leftarrow nova\_carga \times \alpha + cargas[port] \times (1 - \alpha)$

Algoritmo 4 Algoritmo implementando a política **load**

1: **function** LOAD($replicas, cargas$)
2: ▷ $replicas$: conjunto de servidores réplica físicos ordenados pelo IP
3: ▷ $cargas$: carga média calculada em $PortStats$ usando EWMA
4: **return** $\arg\min_i \ (cargas[i])$

A política **load-prev-δ** é similar a **load** ao utilizar *EWMA* para obter a carga média de cada servidor, porém, durante o período entre o recebimento das estatísticas via *PortStats*, acrescentamos uma previsão de fluxo à carga do servidor selecionado pelo controlador, como pode ser visto na linha 6 do Algoritmo 5. Assim, a cada recebimento de

Algoritmo 5 Algoritmo implementando a política **load-prev-**δ

1: **function** LOAD-PREV($replicas, cargas, \delta$)
2: ▷ $replicas$: conjunto de servidores réplica físicos ordenados pelo IP
3: ▷ $cargas$: carga média calculada em $PortStats$ usando EWMA
4: ▷ δ: acréscimo de carga
5: $j \leftarrow \arg\min_i (cargas[i])$
6: $cargas[replicas[j].port] += \delta$ ▷ Acrescenta previsão de carga ao servidor
7: **return** j

novo fluxo, quando o controlador determina qual servidor físico receberá o fluxo, o valor da carga média do servidor selecionado é incrementado por um valor δ, que representa o incremento de carga previsto a cada conexão. Nos experimentos realizados, o δ é fixo e definido na inicialização do controlador. Em trabalhos futuros pretendemos acrescentar um algoritmo de aprendizado para que o incremento seja alterado dinamicamente.

4. Avaliação Experimental

Essa seção foi dividida em duas partes: (1) Montagem do protótipo: descrição do hardware e software utilizado e a metodologia adotada na avaliação; e (2) Resultados e Análise: apresentação e análise dos resultados obtidos nos experimentos. A plataforma física simula um ambiente real similar ao mostrado na Figura 1. A seguir são descritos o hardware e software utilizados.

4.1. Montagem do protótipo

O protótipo consiste de uma estação que gera a carga dos clientes, um switch OpenFlow e quatro servidores. A estação é um computador Intel Core i5-2310 2,9 GHz, 8 GB RAM DDR3 e placa de rede Gigabit Ethernet. Os servidores possuem CPU Intel Core i7-3770 3,4 GHz, 16 GB RAM DDR3 e placa de rede Gigabit Ethernet rodando Apache. Nos experimentos foram utilizados quatro servidores web físicos. O Controlador é um Sony Vaio VGN-C240E com CPU Intel Core2 Duo T5500 1,66 GHz, 2 GB RAM DDR2 e uma placa de rede Ethernet 10/100 Mbps. O switch é um Pica8 P-3297, firmware 2.73, com 24 portas Gigabit Ethernet e OpenFlow 1.0. O Pica8 foi escolhido por ser um switch SDN de alto desempenho, que é empregado em diversas plataformas experimentais de SDN, tais como o FIBRE[4] e o FUTEBOL[5].

O controlador implementa os algoritmos de controle, que executam sobre o POX 0.3.0 (Carp). Na inicialização, os algoritmos limpam a tabela de fluxos do switch. Novos fluxos HTTP são tratados pelo algoritmo de balanceamento de carga, enquanto fluxos de outros tipos são tratados utilizando um switch l2_learning do POX.

As estatísticas de *PortStats* são amostradas em intervalos fixo de $10\ ms$ entre cada disparo. Valores inferiores a $5\ ms$ geram uma sobrecarga no switch Pica8, provocando a desconexão do canal de comunicação de controle entre o controlador e o switch. Para as políticas **txbytes** e **cpuq-load**, a aplicação em *sockets* é executada a cada *PacketIn*, ou seja, o controlador busca a informação de todos os servidores web a cada novo fluxo.

[4]http://www.fibre.org.br
[5]http://www.ict-futebol.org.br

No cliente são registradas quantas requisições foram solicitadas, o tempo médio de cada requisição (tempo de conexão), o tempo total, erros, conexões perdidas e a vazão média. O objetivo é usar o tempo médio de resposta para cada requisição para avaliar as políticas de Engenharia de Tráfego.

Os experimentos utilizaram 20, 60, 80, 100, 120, 160, 180 e 200 conexões simultâneas ao endereço do servidor virtual. Os valores de 140 e 160 conexões simultâneas não foram avaliados por estarem dentro de um intervalo de respostas lineares do switch, conforme pode ser observado entre 100 e 160 conexões em todas as curvas exibidas. Não foi utilizado um maior número de conexões simultâneas porque o switch utilizado apresentou um princípio de queda no desempenho com mais conexões simultâneas. Esses resultados corroboram com os resultados de desempenho de outros switches na literatura [Costa et al. 2016]. Foram realizados 33 experimentos para cada configuração, e os resultados foram avaliados para um intervalo de confiança de 95%. O valor de δ foi ajustado empiricamente para a carga com 20 conexões simultâneas. Verificamos que neste faixa, o melhor valor é $\delta = 13$ KB/s. Este valor é bastante inferior à taxa de transferência da rede que ficou em média 1080 KB/s com 20 conexões simultâneas.

4.2. Carga homogênea

Neste cenário utilizamos o HTTPERF[6] para simular o tráfego de vários clientes. A carga é homogênea, pois cada requisição solicita um arquivo de 100 KB. O tempo de resposta apresentado nas figuras 2, 3, 4 e 5 corresponde ao tempo total medido entre a requisição da URI feita pelo cliente e o recebimento completo do arquivo correspondente.

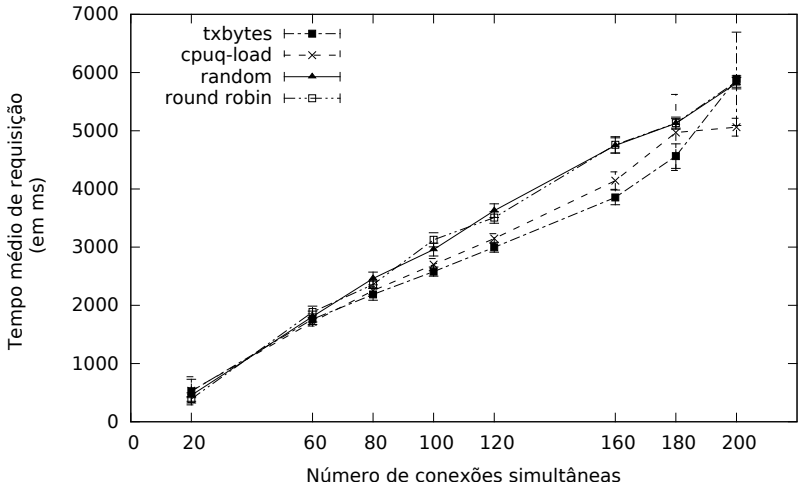


Figura 2. Comparação do *baseline* com políticas que buscam dados via *sockets*.

A Figura 2 mostra uma comparação do tempo médio de resposta utilizando as políticas clássicas de **round robin** e **random** com as políticas **txbytes** e **cpuq-load**, as quais buscam informações diretamente do servidor. Observa-se que com poucas conexões as quatro políticas são estatisticamente iguais. Contudo, à medida em que o número de conexões aumenta, a política **txbytes** sofre um aumento no tempo para tomada de decisão pelo controlador. Apesar disso, há uma melhora no desempenho percebido pelos clientes,

[6]https://github.com/httperf/httperf

o que é refletido na diminuição do tempo de resposta. Essa mesma melhora, porém em menor intensidade, também pode ser percebida com o uso da política **cpuq-load**.

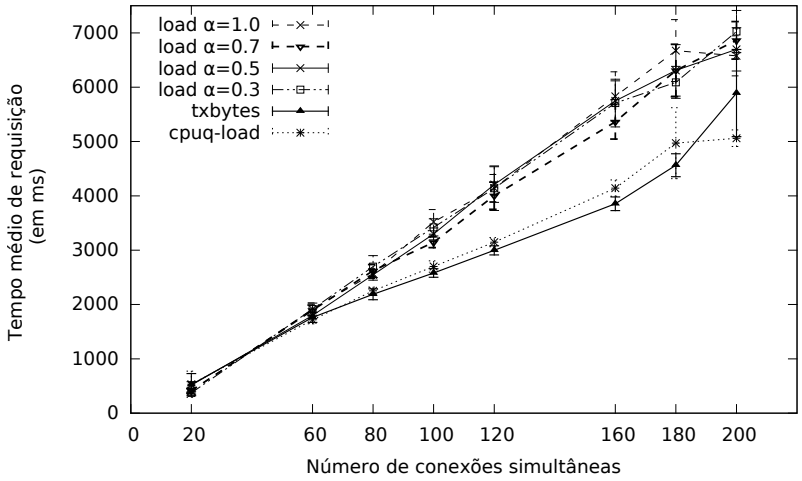


Figura 3. Comparação com política de carga via PortStats.

A política de previsão de carga **load** é comparada com **txbytes** e **cpuq-load** na Figura 3. As política **load** foi avaliada com valores de α variando em 0,3, 0,5, 0,7 e 1. Dessa forma verifica-se uma influência mais importante dos valores anteriores de carga para $\alpha = 0,3$ na carga média. Quando $\alpha = 1$, a carga média desconsidera completamente os resultados passados. Os melhores resultados entre as políticas **load** ocorreu para $\alpha = 0,7$. Isso confirma que a carga deve ser ajustada de forma a considerar mais fortemente o último tráfego medido, que corresponde às novas requisições que devem ser atendidas pelo servidor, entretanto, não é possível desconsiderar a carga medida no passado, pois o servidor web ainda pode estar processando parte dessa carga.

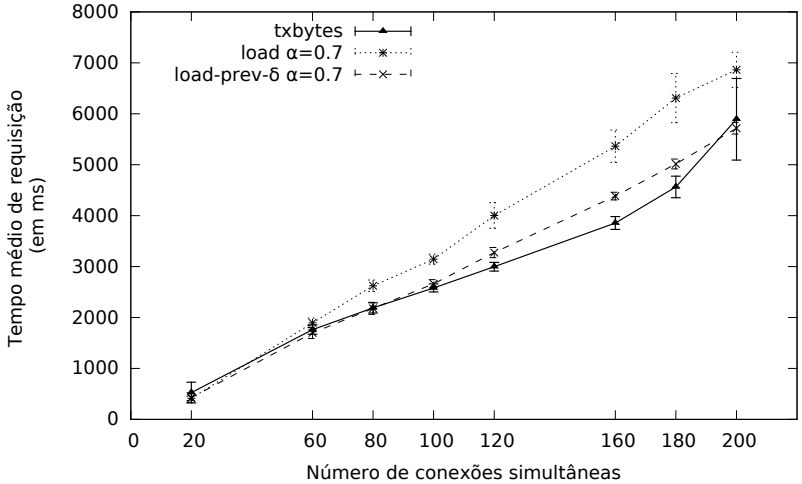
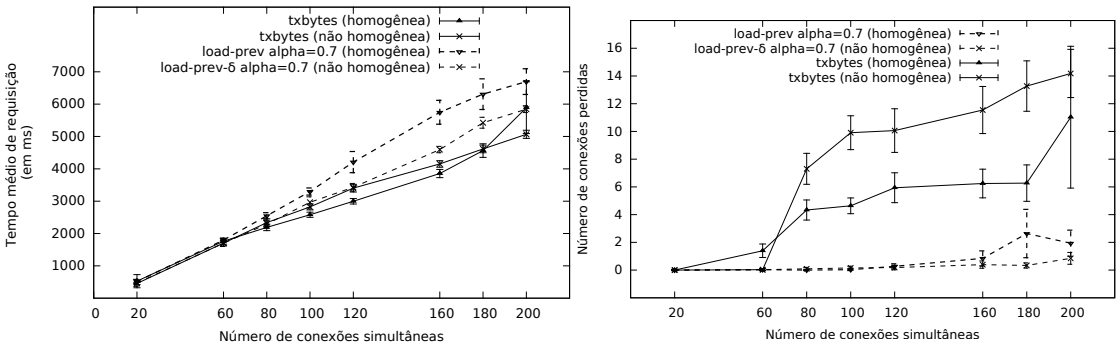


Figura 4. Comparação de políticas de carga com e sem previsão com txbytes.

Na Figura 4 é exibida uma comparação do desempenho da política **load** com **load-prev-**δ, ambas com $\alpha = 0,7$ que foi verificado anteriormente como o melhor valor de α para **load**. Estas duas políticas de previsão de carga são comparadas com **txbytes** que foi política que teve o melhor desempenho. Nota-se que com a previsão, os resultados

de **load-prev-δ** foram melhores que os demais na região próxima onde δ foi ajustado. A partir de 120 conexões, o valor **load-prev-δ** fica pior que **txbytes**, o que pode ser explicado por meio do ajuste do valor de δ para um número de conexões baixo. Para todos os casos avaliados, a política **load-prev-δ** é superior a **load**. Esse resultado era esperado pois **load** não considera a carga adicionada na rede durante o intervalo entre os *PortStats*, enquanto **load-prev-δ** considera o acréscimo de novo fluxo sobre a carga do servidor.



(a) Tempo médio de requisição. (b) Conexões perdidas

Figura 5. Comparação entre políticas com carga homogênea e heterogênea.

4.3. Carga heterogênea

Neste cenário simulamos uma carga heterogênea. Seguimos a metodologia de Summers *et al*. para geração de cargas de vídeo utilizando HTTPERF, baseado na utilização de sessões com *chunks* simulando o funcionamento de *Real-time Transport Protocol* (RTP) e *Real-Time Streaming Protocol* (RTSP) [Summers et al. 2012]. Foram configuradas sessões para o HTTPERF selecionando entre 588 blocos diferentes cujo tamanho varia até 2,3 vezes entre o menor e o maior bloco disponível. Foram criadas cerca de 100 sessões diferentes utilizando estes valores, desta forma são criadas sessões de tamanhos e sequências de requisições diferentes.

Na Figura 5(a) são exibidos os resultados obtidos por **txbytes** e **load-prev** com carga homogênea e com carga não homogênea. É possível observar que **txbytes** obtém os melhores resultados, indicando desta forma que a quantidade de bytes transmitidos é um melhor indicador para o desempenho. Isso se deve ao fato do desempenho do conjunto controlador e switch OpenFlow não ser capaz de atender adequadamente ao número de conexões. É possível verificar na Figura 5(b) que a perda média é zero para 20 conexões simultâneas, contudo, à medida em que o número de conexões simultâneas aumenta, o número de perdas de conexões também aumenta. É importante observar que o número de perdas de conexões para **txbytes** cresce mais rapidamente que em **load-prev**. Desta forma **txbytes** garante que as conexões que podem ser feitas sejam transmitidas pelo switch, obtendo um menor tempo de resposta, porém às custas de uma maior quantidade de perdas de conexão.

4.4. Discussão

O algoritmo que obteve melhor desempenho foi o **txbytes** que utiliza *sockets*. Embora os algoritmos **load** e **load-prev** obtiveram resultados próximos do **txbytes**, era esperado que eles tivessem melhor desempenho.

É esperado que a implementação da aquisição de estatíticas no switch OpenFlow fosse mais eficiente, e portanto mais rápida do que aplicações de sockets em cada servidor réplica. Porém, conforme os resultados pode-se concluir que as implementações físicas ainda devem evoluir para serem mais rápidas nos switches OpenFlow. O algoritmos que utilizam as estatísticas *PorStats*, mesmo não sendo disparados a cada PacketIn, mas em intervalos fixos de 10 *ms*, possuem um atraso devido às estas operações internas do switch. Com isto podemos supor que se a cada PacketIn as estatísticas fossem requisitadas ao switch, maiores atrasos seriam adicionados.

Pelas implementações e resultados obtidos pode-se confirmar a validade das soluções de divisão de tráfego neste trabalho. Novas gerações de dispositivos SDN e do protocolo OpenFlow podem ser esperadas para melhorar o desempenho. Sem dúvida, os switches SDN possuem requisitos que demandam maior desempenho do hardware conforme já publicado na literatura [Curtis et al. 2011], quando comparados com redes em arquiteturas tradicionais não SDN.

5. Conclusão e Trabalhos Futuros

Neste trabalho foram implementadas cinco políticas de divisão de tráfego utilizando uma rede OpenFlow, das quais duas são políticas clássicas – **round robin** e **random**, para fins de comparação e linha base (*baseline*). As políticas de previsão de carga baseadas na leitura de estatísticas do switch, via primitiva *PortStats* do OpenFlow, têm um pior desempenho do que a linha base devido ao atraso de execução desta primitiva. Embora haja uma melhora com a política de previsão load-prev-δ, o melhor valor para δ dependerá de características do hardware de cada switch e do perfil do tráfego, que não é conhecido previamente. As políticas cpuq-load e txbytes, que executam a leitura de carga diretamente nos servidores via sockets, são as que resultam em melhor desempenho. A política txbytes é melhor do que a cpuq-load por indicar valor mais significativo da carga e com os menores atrasos.

A implementação do protocolo OpenFlow nos switches requer maior rapidez computacional em várias ações, antes inexistentes nos switches tradicionais tal como se verifica, por exemplo, na leitura das estatísticas por fluxo e inserção de regras. As informações gravadas para estas ações ainda devem ser executadas num período de tempo menor do que o atingido pelo switch físico utilizado nos experimentos. Acreditamos que com o desenvolvimento de novas gerações de switches OpenFlow, será possível uma leitura das estatísticas em tempo suficiente para que possam ser utilizadas na política que é esperada ser a mais rápida dentre todas deste trabalho, a política load.

Como trabalhos futuros, pretendemos avaliar refinamentos às soluções apresentadas. Por exemplo, implementar políticas em que não utilizem reescrita de cabeçalho para reduzir o atraso, bem como empregar valores de δ variáveis. Além disso, novas plataformas SDN, tais como a PISA - *Protocol Independent Switch Architecture* - da empresa Barefoot Networks[7], serão avaliadas.

Referências

[Akyildiz et al. 2003] Akyildiz, I. F., Anjali, T., Chen, L., De Oliveira, J. C., Scoglio, C., Sciuto, A., Smith, J. A., and Uhl, G. (2003). A New Traffic Engineering Manager

[7] https://www.barefootnetworks.com/

for DiffServ/MPLS Networks: Design and Implementation on an IP QoS Testbed. *Comput. Commun.*, 26(4):388–403.

[Al-Fares et al. 2008] Al-Fares, M., Loukissas, A., and Vahdat, A. (2008). A Scalable, Commodity Data Center Network Architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74.

[Armbrust et al. 2010] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A View of Cloud Computing. *Commun. ACM*, 53(4):50–58.

[Cardellini et al. 2002] Cardellini, V., Casalicchio, E., Colajanni, M., and Yu, P. S. (2002). The state of the art in locally distributed web-server systems. *ACM Computing Surveys (CSUR)*, 34(2):263–311.

[Costa et al. 2016] Costa, L. C., Vieira, A. B., Silva, E. B., Macedo, D. F., Gomes, G., Correia, L. H., and Vieira, L. F. (2016). Avaliação de desempenho de plano de dados openflow. In *XXXIV Simpósio Brasileiro de Rede de Computadores, 2016*. SBC.

[Curtis et al. 2011] Curtis, A. R., Mogul, J. C., Tourrilhes, J., Yalagandula, P., Sharma, P., and Banerjee, S. (2011). Devoflow: Scaling flow management for high-performance networks. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, pages 254–265, New York, NY, USA. ACM.

[Ferraris et al. 2012] Ferraris, F., Franceschelli, D., Gioiosa, M., Lucia, D., Ardagna, D., Di Nitto, E., and Sharif, T. (2012). Evaluating the Auto Scaling Performance of Flexiscale and Amazon EC2 Clouds. In *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 423–429.

[Gandhi et al. 2014] Gandhi, R., Liu, H. H., Hu, Y. C., Lu, G., Padhye, J., Yuan, L., and Zhang, M. (2014). Duet: Cloud Scale Load Balancing with Hardware and Software. In *ACM SIGCOMM*, pages 27–38.

[Guedes et al. 2012] Guedes, D., Vieira, L. F. M., Vieira, M. M., Rodrigues, H., and Nunes, R. V. (2012). Redes definidas por software: uma abordagem sistêmica para o desenvolvimento de pesquisas em redes de computadores. *Minicursos do Simpósio Brasileiro de Redes de Computadores*, 30(4):160–210.

[Guo et al. 2014] Guo, Z., Su, M., Xu, Y., Duan, Z., Wang, L., Hui, S., and Chao, H. J. (2014). Improving the performance of load balancing in software-defined networks through load variance-based synchronization. *Computer Networks*, 68:95–109.

[Handigol et al. 2009] Handigol, N., Seetharaman, S., Flajslik, M., and Johari, R. (2009). Plug-n-Serve: Load-balancing web traffic using OpenFlow. *Demo at ACM SIGCOMM*.

[Jain et al. 2013] Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Hölzle, U., Stuart, S., and Vahdat, A. (2013). B4: Experience with a Globally-deployed Software Defined Wan. *SIGCOMM Comput. Commun. Rev.*, 43(4):3–14.

[Leduc et al. 2006] Leduc, G., Abrahamsson, H., Balon, S., Bessler, S., D'Arienzo, M., Delcourt, O., Domingo-Pascual, J., Cerav-Erbas, S., Gojmerac, I., Masip, X., Pescapè, A., Quoitin, B., Romano, S., Salvadori, E., Skivée, F., Tran, H., Uhlig, S., and Ümit,

H. (2006). An Open Source Traffic Engineering Toolbox. *Computer Communications*, 29(5):593 – 610.

[Macedo et al. 2015] Macedo, D. F., Guedes, D., Vieira, L. F. M., Vieira, M. A. M., and Nogueira, M. (2015). Programmable networks: From software-defined radio to software-defined networking. *IEEE Communications Surveys Tutorials*, 17(2):1102–1125.

[Patel et al. 2013] Patel, P., Bansal, D., Yuan, L., Murthy, A., Greenberg, A., Maltz, D. A., Kern, R., Kumar, H., Zikos, M., Wu, H., et al. (2013). Ananta: cloud scale load balancing. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 207–218.

[Pfaff et al. 2015] Pfaff, B., Pettit, J., Koponen, T., Jackson, E., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P., Amidon, K., and Casado, M. (2015). The design and implementation of Open vSwitch. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 117–130.

[Rodrigues et al. 2015] Rodrigues, C. P., Costa, L. C., Vieira, M. A. M., Vieira, L. F. M., Macedo, D. F., and Vieira, A. B. (2015). Avaliação de balanceamento de carga web em redes definidas por software. In *XXXIII Simpósio Brasileiro de Rede de Computadores, 2015*, pages 585–597. SBC.

[Summers et al. 2012] Summers, J., Brecht, T., Eager, D., and Wong, B. (2012). Methodologies for generating http streaming video workloads to evaluate web server performance. In *International Systems and Storage Conference (SYSTOR)*, pages 2:1–2:12.

[Wang et al. 2011] Wang, R., Butnariu, D., and Rexford, J. (2011). OpenFlow-based Server Load Balancing Gone Wild. In *USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, pages 12–12.

[Xu et al. 2014] Xu, D., Fan, B., and Liu, X. (2014). Efficient Server Provisioning and Offloading Policies for Internet Datacenters With Dynamic Load-Demand. *IEEE Transactions on Computers*, 99:1.

Trilha Principal do SBRC 2017
Sessão Técnica 3
Sistemas Distribuídos

Além da Escalabilidade: Inteligência de Enxames Afetada por Campos Magnéticos em Espaços de Tuplas Distribuídos

Henrique Duarte Lima[1], **Luiz A. de P. Lima Jr.**[1], **Alcides Calsavara**[1], **Henri F. Eberspächer**[1], **Ricardo C. Nabhen**[1], **Elias P. Duarte Jr**[2]

[1] Programa de Pós-Graduação em Informática (PPGIa)
Pontifícia Universidade Católica do Paraná (PUCPR)

[2]Departamento de Informática - Universidade Federal do Paraná (UFPR)

henrique@engduarte.com, {laplima,alcides}@ppgia.pucpr.br,

{henri.eberspacher,ricardo.nabhen}@pucpr.br, elias@inf.ufpr.br

Abstract. Tuple Spaces *simplify the process of creating distributed systems providing a model for component communication and synchronization that is loosely copled in terms of both space and time. However, traditional implementations in high-performance environments present low scalability. Although "anti-over-clustering" techniques allow the construction of more scalable solutions, the excessive dispersion of elements throughout the network compromises the overall performance. In this work we propose a strategy to cope with this problem:* Magnetic SwarmLinda*, an approach based on swarm intelligence and on the concept of virtual magnetic fields. Simulation results show that the proposed strategy significantly improves the system's performance particularly the tuple retrieval latency.*

Resumo. Espaços de Tuplas *simplificam a construção de sistemas distribuídos provendo um modelo de comunicação e sincronização de componentes com desacoplamento espacial e temporal. Porém, implementações tradicionais em ambientes com demandas de alto desempenho exibem baixa escalabilidade. Embora técnicas de "*anti-over-clustering*" permitam a construção de soluções mais escaláveis, mostramos que o espalhamento excessivo resultante reduz o desempenho do sistema. Para lidar com o problema, o* Magnetic SwarmLinda*, um método baseado em inteligência de enxames e no conceito de campos magnéticos virtuais, é apresentado. Resultados de simulação confirmam que a estratégia proposta melhora efetivamente o desempenho do sistema em termos da latência da recuperação de tuplas.*

1. Introdução

Os Espaços de Tuplas Distribuídos – ou simplesmente Espaços de Tuplas [Atkinson 2008] – são um modelo baseado em memória associativa compartilhada para comunicação e coordenação de processos em sistemas paralelos e distribuídos. Os dados são armazenados no espaço compartilhado na forma de *tuplas*, que são listas ordenadas de dados de diferentes tipos. A tupla $\langle \text{“}SBRC\text{”}, 2017 \rangle$, por exemplo, contém valores para dois elementos, o primeiro do tipo *string* e o segundo, do tipo número inteiro. Processos clientes de um Espaço de Tuplas Distribuído são *produtores* e/ou *consumidores* de tuplas.

A diferença fundamental entre os Espaços de Tuplas e outros modelos consiste no fato das tuplas nunca serem explicitamente endereçadas. A busca de tuplas no Espaço de Tuplas é feita de forma associativa, através de *templates*, que especificam um "formato" da tupla, bem como restrições de valores (por exemplo, o *template* $\langle [string], 2017 \rangle$ representa todas as tuplas com 2 elementos, sendo o primeiro uma *string*, e o segundo o valor 2017). Outra característica importante é o fato de que processos que utilizam um Espaço de Tuplas não interagem diretamente entre si: toda comunicação acontece através da inserção e busca/remoção de tuplas no Espaço de Tuplas. Por causa desta característica, há um baixo acoplamento de processos no espaço e no tempo. No espaço, porque processos podem estar espalhados de forma arbitrária sobre uma rede de computadores. No tempo, porque processos não precisam estar executando no mesmo instante de tempo para poderem se comunicar – um processo pode criar e inserir uma tupla no Espaço de Tuplas para, logo em seguida, encerrar sua execução. Outro processo mais tarde pode iniciar a sua execução e consumir aquela tupla armazenada.

O modelo de Espaço de Tuplas foi originalmente proposto e implementado no contexto da linguagem Linda [Gelernter and Bernstein 1982] e é, portanto, frequentemente associado à própria linguagem. As duas operações básicas para inserir e recuperar tuplas no Espaço de Tuplas são `out` e `in`, respectivamente. A operação `in` para um determinado *template* remove do Espaço de Tuplas a primeira tupla encontrada (em uma escolha não determinística) que possua o formato especificado pelo *template*. Caso não haja no Espaço de Tuplas nenhuma tupla do formato solicitado, o processo solicitante fica bloqueado até que uma tupla apropriada seja posteriormente publicada por meio da operação `out`.

Apesar de terem seus méritos reconhecidos, os Espaços de Tuplas originais apresentam problemas de escalabilidade em ambientes de alto desempenho. Nos sistemas originais, e inclusive na implementação JavaSpaces [Waldo et al. 1997], um componente central é responsável pelo processamento que antecede tanto a inserção como a recuperação de tuplas. Diversos esforços (descritos na Seção 2 deste trabalho) foram realizados para eliminar este componente centralizado e melhorar o desempenho sobretudo da operação `in`. Estas abordagens distribuídas tornaram possível a aplicação dos Espaços de Tuplas em novas áreas como [Maia et al. 2016] a *Internet of Coisas* (IoT) (na comunicação com baixo nível de acoplamento entre sensores, atuadores e aplicações) e aplicações que executam em nuvens [Hari 2012].

Uma das abordagens descentralizadas com potencial para tais utilizações é o *SwarmLinda* [Menezes and Tolksdorf 2003] que é uma implementação baseada em inteligência de enxames. No *SwarmLinda*, sempre que uma operação `out` é executada, uma formiga é instanciada. Essa formiga, denominada de *tuple-ant*, é responsável por percorrer os nós que compõem o Espaço de Tuplas para encontrar um nó adequado para depositar a tupla. O critério utilizado para depositar uma tupla é a quantidade de tuplas "semelhantes" à tupla carregada pela formiga que já estão depositadas no nó atual. Isto permite a formação de *clusters* de tuplas similares com o objetivo de acelerar buscas posteriores. A estratégia, entretanto, não resolve o problema da escalabilidade, pois conforme a quantidade de tuplas semelhantes depositadas aumenta, ocorre uma degradação importante do desempenho devido a concentração de processamento em certos nós. Além disso, nós sobrecarregados acabam tornando-se mais atrativos por terem uma alta concentração

de tuplas similares – e a situação tende a piorar com o passar do tempo.

Em [Casadei et al. 2007] é proposta a abordagem *Anti-Over-Clustering* para evitar a sobrecarga de nós devido à concentração excessiva de tuplas. Nesta abordagem, a probabilidade de uma formiga depositar uma tupla também aumenta de acordo com a quantidade de tuplas similares. No entanto, quando a quantidade de tuplas aproxima-se de um limiar, esta probabilidade diminui ainda que a quantidade de tuplas aumente. Desta forma, as *tuple-ants* evitam depositar suas tuplas em nós sobrecarregados. Infelizmente, esta estratégia obriga as formigas a continuarem a exploração do Espaço de Tuplas, produzindo um maior consumo de recursos de processamento e comunicação do sistema. Além disso, a estratégia leva à formação de *clusters* dispersos no Espaço de Tuplas, o que, no balanço final, reduz sensivelmente o desempenho da recuperação de tuplas.

Este trabalho descreve o *Magnetic SwarmLinda*: uma estratégia baseada em inteligência de enxames afetada por "campos magnéticos virtuais" para resolver o problema da dispersão de *clusters* distribuídos de forma a manter a escalabilidade e eficiência na recuperação de tuplas. Campos Magnéticos Virtuais implementam um modelo distribuído autonômico de relações de auxílio entre nós formando redes *overlay* para o encaminhamento de mensagens de aplicação aos nós mais aptos (i.e., aqueles com maior força de atração magnética). A intensidade do campo magnético é parâmetro dinâmico dependente da aplicação. O modelo foi originalmente proposto para o encaminhamento de mensagens a destinatários móveis e para implementar mecanismos distribuídos de balanceamento de carga [de Paula Lima Jr. and Calsavara 2010] (neste caso, quanto maior a ociosidade de um nó, maior a sua força de atração). Na abordagem proposta no presente trabalho, campos magnéticos atuam como "campos de força" que produzem um comportamento "anormal" nas formigas, protegendo cada nó de uma sobrecarga resultante de uma concentração excessiva de tuplas similares. O *Magnetic SwarmLinda* foi implementado e os resultados obtidos confirmam que a formação de *clusters* de *clusters* (grupos de nós vizinhos contento tuplas similares) resultantes do método proposto produz um aumento considerável do desempenho global do sistema em comparação com todas as abordagens concorrentes.

O restante do trabalho está organizado da seguinte maneira. Na Seção 2, são apresentados trabalhos relacionados, inclusive os campos magnéticos virtuais. Na Seção 3, o *Magnetic SwarmLinda* é descrito. Resultados experimentais são apresentados na Seção 4, comparando os benefícios da estratégia proposta com o concorrente mais direto, o *Anti-Over-Clustering*. As conclusões seguem na Seção 5.

2. Trabalhos Relacionados

Nesta Seção, serão apresentadas as principais abordagens *distribuídas* de Espaços de Tuplas. Iniciamos pelo WCL [Rowstron 1998], que é uma implementação de múltiplos Espaços de Tuplas distribuídos geograficamente. Cada Espaço de Tuplas é executado sobre um único nó, sendo, portanto, centralizado. No entanto, o WCL é capaz de realizar a migração de Espaços de Tuplas de forma transparente entre os nós que compõem o sistema. A migração pode ocorrer com o intuito de reduzir a latência de comunicação com um determinado Espaço de Tuplas, migrando este para um nó de onde provém a maior parte das requisições. Além disso, a migração pode ocorrer quando a carga em um determinado nó está elevada, permitindo o balanceamento de carga. No entanto, esta

estratégia é eficaz apenas quando a sobrecarga do nó é resultado da soma das cargas de vários Espaços de Tuplas. Quando um determinado Espaço de Tuplas é capaz de sobrecarregar sozinho um nó, o balanceamento de carga torna-se inócuo.

Linda in Mobile Environment (LIME) [Picco et al. 1999] é uma implementação de Espaço de Tuplas que permite que agentes móveis transitem entre dispositivos móveis heterogêneos. Cada agente tem uma visão do sistema em termos do conjunto de tuplas disponíveis e esta modifica-se, de maneira transparente, conforme ocorrem mudanças de conectividade entre os *hosts* dos demais agentes. É possível também que o volume de tuplas se altere em função da movimentação de agentes entre os *hosts*. A distribuição das tuplas no LIME ocorre de maneira diferenciada. Cada tupla adicionada no Espaço de Tuplas é armazenada no *host* executando o agente responsável pela inserção desta tupla. Quando um agente migra de *host*, este carrega consigo todas as suas tuplas que ainda não foram recuperadas. Devido a este estilo de distribuição das tuplas, quando um agente não consegue encontrar uma tupla qualquer na sua porção de um Espaço de Tuplas compartilhado com outros agentes, este agente precisa verificar os demais nós do Espaço de Tuplas até encontrar a tupla desejada, o que envolve atrasos sem limite conhecido.

O *Dtuples* [Jiang et al. 2006] é uma implementação de Espaço de Tuplas construída sobre uma tabela de *hash* distribuída com o objetivo de simplificar a comunicação entre agentes em ambientes distribuídos. Há uma restrição imposta pelo Dtuples, na qual o elemento da posição inicial da tupla deve ser uma *string* com o nome da tupla, sendo que este valor não necessita ser único. Esta restrição deve-se ao fato da implementação utilizar o *hash* deste primeiro elemento para determinar o local de armazenamento da tupla. Na recuperação de uma tupla também é necessário informar este nome no primeiro campo do *template*, permitindo que o Espaço de Tuplas saiba em qual local deve ser executado o processo de busca. Infelizmente, esta abordagem permite uma concentração indesejável de carga em poucos nós quando há muitas colisões de *hash*. Além disso, essa restrição impõe mudanças na interface tradicional dos Espaços de Tuplas.

O *Tupleware* [Atkinson 2008] é um *middleware* inspirado em Linda que implementa um Espaço de Tuplas que dispõe de um algoritmo descentralizado para recuperação de tuplas. As tuplas inseridas por um processo são armazenadas em uma instância local que compõe o Espaço de Tuplas, permitindo assim uma concentração excessiva de tuplas em apenas alguns nós, dependendo das características de comunicação da aplicação. Quando um processo requisita uma determinada tupla que não está disponível localmente, é efetuada uma busca - de forma transparente para a aplicação - nos demais nós que compõem o Espaço de Tuplas. A busca nos demais nós ocorre de maneira sequencial, sendo que os primeiros nós a serem verificados são aqueles que apresentam um elevado *fator de sucesso*. Este valor indica a taxa histórica de sucesso produzida pelas buscas em determinado nó, sendo incrementado em caso de sucesso e decrementado em cenário oposto. Desta forma, nós que eventualmente possuam uma grande porção das tuplas do Espaço de Tuplas serão, provavelmente, muito consultados pelos demais nós, provocando uma distribuição inadequada da carga do Espaço de Tuplas.

O *SwarmLinda* [Menezes and Tolksdorf 2003] é uma implementação de Espaço de Tuplas que usa técnicas de inteligência de enxames. Nessa abordagem, as características do Espaço de Tuplas emergem das interações entre indivíduos simples, inspirados pelas formigas, que cooperam de forma descentralizada. As decisões tomadas por

estes indivíduos baseiam-se unicamente em informações locais, tornando desnecessário o conhecimento do estado global do sistema, evitando assim uma troca intensa de mensagens. A estratégia de movimentação das formigas utiliza como critérios a intensidade do feromônio e a concentração local de tuplas semelhantes. Quando uma formiga não pode atingir seu objetivo no nó atual, ela obtém informações sobre o número de tuplas semelhantes e a intensidade do feromônio em nós vizinhos para determinar a probabilidade de se mover para cada um desses nós.

Uma formiga é instanciada sempre que uma operação `out` é executada. Essa formiga, denominada de *tuple-ant*, é responsável por percorrer os nós do Espaço de Tuplas para encontrar um nó adequado para depositar a tupla. O critério utilizado para depositar uma tupla é o número de tuplas depositadas no nó que são similares à tupla carregada pela formiga, permitindo a formação de "*clusters*" de tuplas similares. Durante seu percurso, a *tuple-ant* deposita um feromônio, que é específico para o tipo de tupla, em cada nó que é visitado. Além disso, o feromônio também é espalhado nos vizinhos do nó onde a tupla é depositada. Similar à operação `out`, sempre que uma operação `in` uma formiga é instanciada. Esta formiga, que é denominada de *template-ant*, é responsável por percorrer os nós, a fim de encontrar uma tupla compatível com seu *template*. Quando a *template-ant* encontra uma tupla compatível, ela deve retornar ao seu "local de nascimento" carregando a tupla enquanto sinaliza o percurso com feromônio relativo à tupla em questão.

A formação dos *clusters* de tuplas similares permite que as formigas não necessitem percorrer todo o Espaço de Tuplas em busca de seus objetivos. Ao invés disto, cada formiga deve concentrar seus esforços em localizar um caminho, através das trilhas formadas por feromônio, que leve até um *cluster* de tuplas de seu interesse. Neste local, a probabilidade da formiga obter sucesso é maior que nos demais nós. No entanto, conforme a quantidade de tuplas depositas nos *clusters* aumenta, ocorre uma degradação do desempenho devido ao excesso de formigas que precisam ser processadas nesses nós. Além disso, nós sobrecarregados acabam tornando-se mais atrativos – na perspectiva das formigas – devido a quantidades elevadas de tuplas e feromônio, provocando a formação de um "círculo vicioso" que provoca um aumento crescente da quantidade de tuplas.

[Casadei et al. 2007] propõe a abordagem *A*nti-Over-Clustering para evitar a sobrecarga de nós devido à concentração excessiva de tuplas. Neste modelo, a probabilidade de depositar uma tupla aumenta de acordo com a quantidade de tuplas similares. No entanto, a partir de um momento em que a quantidade de tuplas aproxima-se de um *threshold*, a probabilidade diminui ainda que a quantidade de tuplas aumente. Desta forma, as *tuple-ants* evitam de depositar suas tuplas em nós sobrecarregados. Infelizmente, esta estratégia obriga que essas formigas continuem a explorar o Espaço de Tuplas, produzindo uma maior carga de processamento sobre o sistema. Além disso, é propiciada a formação de *clusters* similares excessivamente dispersos no Espaço de Tuplas, dificultando a recuperação de tuplas.

3. Proposta: *Magnetic SwarmLinda*

Neste trabalho é apresentada uma estratégia distribuída para a implementação de Espaços de Tuplas Distribuídos denominada *Magnetic SwarmLinda*. A estratégia propõe que o uso da inteligência de enxames seja afetado por Campos Magnéticos Virtuais. O objetivo é prevenir a concentração excessiva de tuplas em um único nó, que pode degradar conside-

ravelmente o desempenho do processo de recuperação de tuplas e, ao mesmo tempo, evitar a dispersão excessiva de tuplas semelhantes, prejudicando o desempenho da recuperação.

A rede é representada por um grafo $G = (V, E)$, sendo V o conjunto de nós e E é o conjunto de arestas bidirecionais entre um par de nós. $N = |V|$ é o número de nós e $M = |E|$ é o número de arestas de G. $NH(i)$ é definido como o conjunto de vizinhos diretos do nó $i \in V$ (i.e., $NH(i) = \{j : (i, j) \in E \vee (j, i) \in E\}$.

As próximas seções detalham o modelo proposto especificando o comportamento das *tuple-ants*, bem como o suporte ao processo de tomada de decisão para depositar/recuperar tuplas.

3.1. Comportamento das *tuple-ants*

Quando um nó recebe uma requisição para executar a operação `out`, os seguintes passos são executados:

1. O nó atribui a tupla informada a um novo agente *tuple-ant* que é responsável por depositar a tupla em algum nó do Espaço de Tuplas. Neste momento, o seu tempo de vida máximo (*Time to Live* - TTL) lhe é atribuído em número de saltos.
2. Em seguida, a formiga verifica se há muitas tuplas depositadas no nó corrente que são similares à tupla carregada. Esta informação é utilizada para decidir se a tupla deve ser depositada. A probabilidade de depositar a tupla aumenta à medida que o número de tuplas similares cresce (conforme será descrito na Seção 3.2).
3. Se a formiga tiver depositado a tupla, sua última tarefa antes de morrer é sinalizar o local através da dispersão de feromônio no nó corrente e nos nós vizinhos. Este processo reforça o nó como um local adequado para tuplas similares, permitindo que outras formigas interessadas nesse tipo de tupla possam encontrar este nó.
4. Se a formiga decidir não depositar a tupla no nó corrente, esta deve escolher um nó adjacente para visitar. Esta escolha é feita estocasticamente com base em informações sobre os nós vizinhos (detalhado na Seção 3.3). No entanto, é possível que esta etapa não ocorra devido à ocorrência de um comportamento "estranho" (que será apresentado na Seção 3.5).
5. Como a formiga se move para um novo nó, esta se torna mais velha, i.e., seu TTL é decrementado. Quando TTL atinge zero, a formiga deposita a sua tupla no nó corrente e dispersa feromônio referente ao tipo da tupla (como descrito na etapa 3) antes de morrer. Caso contrário, a formiga continua a partir da etapa 2.

Este comportamento das *tuple-ants* pode ser simplificado dependendo da equação que define a probabilidade de depositar uma tupla. A verificação do TTL pode ser omitida (conforme o fluxograma da Figura 1) caso a equação que define a probabilidade de depositar uma tupla garanta que a tupla será depositada quando o TTL atingir zero. No modelo proposto, a equação que define a probabilidade de depositar uma tupla (apresentada na Seção 3.2) garante que toda tupla será depositada quando o TTL da formiga responsável pela tupla atingir o valor zero, permitindo tal simplificação.

A decisão de depositar uma tupla baseada na concentração de tuplas ocorre de forma probabilística. A partir da concentração de tuplas similares, a probabilidade $P_{drop}(i, \tau_c)$ de depositar a tupla τ_c no nó atual i é calculada (ver Seção 3.2). Na sequência, a formiga sorteia um número r ($0 \leq r < 1$) e o compara com a probabilidade calculada. Se r for inferior à $P_{drop}(i, \tau_c)$, a formiga deposita a tupla e espalha feromônio referente ao tipo da tupla. Caso contrário, a formiga continua a exploração do Espaço de Tuplas

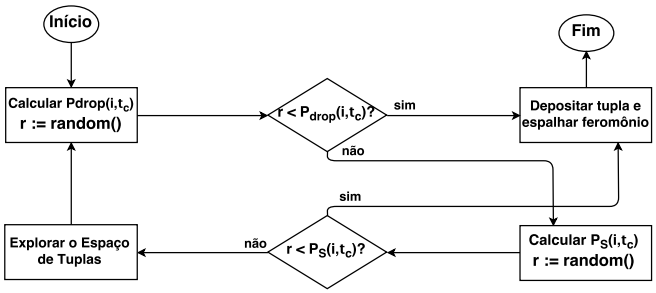


Figura 1. Comportamento simplificado das *tuple-ants*.

(descrito na Seção 3.3) desde que não adote um comportamento estranho (descrito na Seção 3.5). Assim como ocorre na avaliação do $P_{drop}(i, \tau_c)$, a análise da probabilidade de comportamento estranho $P_S(i, \tau_c)$ é realizada comparando o valor da probabilidade com um número aleatório r ($0 \leq r < 1$).

3.2. Probabilidade de Depositar uma Tupla

A probabilidade de depositar uma tupla no nó corrente depende da concentração de tuplas similares no referido nó. A *função de similaridade* – designada por $sim(\tau_A, \tau_B) \in [0, 1]$ – deve ser definida de modo que o grau de similaridade entre duas tuplas τ_A e τ_B possa ser avaliado. Para os experimentos descritos na Seção 4, tuplas são consideradas similares se suas quantidades de campos e os respectivos tipos de cada campo são idênticos.

A concentração de tuplas em determinado nó i de tuplas que são similares à τ_c (a tupla carregada) – designada por $C(i, \tau_c)$ – é dada pela Equação (1). $C(i, \tau_c)$ é determinada através da comparação de τ_c com cada tupla τ_s armazenada em $i \in V$. Caso não exista nenhuma tupla similar à τ_c no nó i, $C(i, \tau_c)$ assumi um valor mínimo para permitir uma pequena probabilidade da tupla τ_c ser depositada em um nó i. Nos experimentos descritos na Seção 4, o valor mínimo de $C(i, \tau_c)$ é 0,0001.

$$C(i, \tau_c) = \sum_{\forall \tau_s \in i} sim(\tau_c, \tau_s) \tag{1}$$

A probabilidade de uma formiga depositar sua tupla τ_c em algum nó i – designada por $P_{drop}(i, \tau_c)$ – é dada pela Equação (2). Observe que $P_{drop}(i, \tau_c)$ depende de TTL.

$$P_{drop}(i, \tau_c) = \left(\frac{C(i, \tau_c)}{C(i, \tau_c) + TTL} \right)^2 \tag{2}$$

Claramente, a probabilidade de depositar a tupla aumenta conforme o TTL diminui. Consequentemente, o TTL representa o limite superior de saltos que uma formiga percorre com o objetivo de depositar sua tupla. Além disso, a probabilidade de depositar uma tupla é maior quando a concentração de tuplas similares é elevada, pois a influência de TTL em $P_{drop}(i, \tau_c)$ diminui, permitindo a formação de *clusters* de tuplas similares.

3.3. Movimentação das Formigas

Se uma formiga não atingir seu objetivo no nó atual, ela precisa escolher um nó vizinho para visitar com o intuito de continuar tentando atingir seu objetivo. A fim de aumentar

suas chances, ela deve mover-se em direção ao local que muitos outros indivíduos portadores de tuplas similares foram. Além do mais, a formiga deve considerar o número de tuplas similares no potencial nó de destino. A Equação (3) define a probabilidade de uma formiga em um nó i carregando uma tupla τ_c de mover-se para um outro nó j, onde $Ph(i, \tau_c)$ representa a quantidade atual de feromônio de tipo τ_c presente no nó $i \in V$.

$$P(\tau_c)_{i,j} = \frac{C(j, \tau_c) + Ph(j, \tau_c)}{\sum_{\forall n \in NH(i)} (C(n, \tau_c) + Ph(n, \tau_c))} \tag{3}$$

Conforme o diagrama da Figura 2, a escolha do próximo nó a ser visitado depende de um valor aleatório r ($0 \leq r < 1$). A formiga percorre o vetor $NH(i)$, que contém os nós adjacentes do nó corrente i, comparando o valor de r com a soma P da probabilidade $P(\tau_c)_{i,j}$ do nó que está sendo avaliado ($index$ atual) e das probabilidades que já foram avaliadas. O nó escolhido é determinado pela iteração no qual o valor de P atinge um valor igual ou superior ao valor de r. Se $index$, que é utilizado como índice para avaliar os vizinhos de i, atingir o valor máximo ($|NH(i)| - 1$), o valor de P atingirá 1. Desta forma, não há a possibilidade da formiga não escolher o próximo nó a ser visitado.

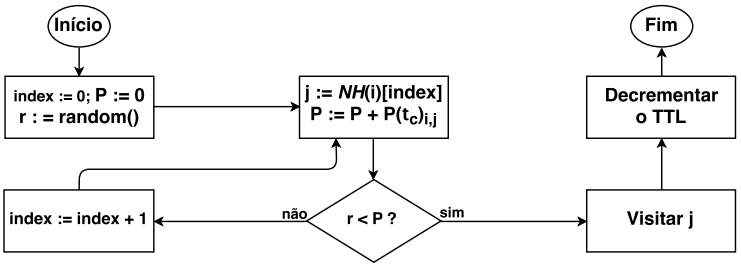


Figura 2. Procedimento de exploração do Espaço de Tuplas.

Quando a formiga encontra o nó j que satisfaz a condição $r < P$, esta desloca-se para o nó j. Além disso, a formiga decrementa o seu TTL, ou seja, a formiga envelhece.

3.4. Evaporação de Feromônio

A evaporação de feromônio é uma mecanismo essencial para tornar o sistema adaptável, uma vez que trilhas referentes a regiões que já não tem uma concentração significante de tuplas devem desaparecer. Um mecanismo de "evaporação" previne que agentes tenham um comportamento caótico que comprometeria o funcionamento adequado do sistema. Além disso, o mecanismo de evaporação torna os caminhos curtos mais atraentes, otimizando assim a quantidade de saltos necessários para que um agente alcance as regiões desejadas do grafo. A Equação (4) define como os feromônios de cada nó i desaparecem conforme o tempo t avança.

$$Ph_t(i, \tau_c) = Ph_{(t-1)}(i, \tau_c)(1 - \rho) \tag{4}$$

Todos os nós do sistema decrementam as suas quantidades de feromônio de acordo com a taxa de evaporação ρ ($\rho \in [0, 1]$). É importante destacar que ρ não deve ser elevado ao ponto de provocar que novas trilhas nunca sejam exploradas. Se ρ é muito baixo, o deslocamento dos indivíduos pode ser afetado negativamente por trilhas que levam à regiões que deixaram de possuir grandes quantidades de tuplas similares.

Apesar de que o uso da inteligência de enxames traz diversos benefícios, um efeito negativo é a sobrecarga dos nós. Embora as trilhas de feromônio constituam um importante mecanismo de orientação das formigas na natureza, este não é o único mecanismo usado por elas. De acordo com [Wajnberg et al. 2010], há também pontos de referência, vibrações, gravidade, bússola solar e luz polarizada que são usadas como critérios de orientação. Em particular, há a influência de campos magnéticos, que, como afirmado por [Banks and Srygley 2003], são capazes de causar uma mudança na orientação de certas espécies de formigas.

[de Paula Lima Jr. and Calsavara 2010] introduz o conceito de campos magnéticos virtuais inspirados da física, para implementar um mecanismo de balanceamento de carga. Neste caso, a intensidade de um campo magnético virtual representa a potência computacional ociosa. Ela é responsável por atrair para os nós mais disponíveis as tarefas para serem processadas. Neste trabalho, a interferência magnética no comportamento das formigas será usada para proteger os nós de forma inteligente de uma sobrecarga que comprometeria o desempenho do sistema.

3.5. Interferência Magnética

Interferência Magnética é um mecanismo cujo objetivo é evitar a concentração de tuplas em poucos (possivelmente sobrecarregados) nós. A inteligência de enxames por si só determina que a probabilidade de um nó receber uma nova tupla é proporcional ao número de tuplas semelhantes que este detém. Esta propriedade melhora o desempenho das subsequentes operações de recuperação de tupla, pois permite que o processo de busca ocorra orientado a apenas uma determinada região do Espaço de Tuplas. No entanto, quando há uma excessiva quantidade de tuplas em um único nó, é altamente provável que a capacidade de processamento do nó torne-se um gargalo.

A sobrecarga de processamento está relacionada ao fato que um nó contendo muitas tuplas provavelmente receberá uma elevada quantidade de formigas tentando recuperar e/ou depositar tuplas. Isto é particularmente grave para *template-ants* que naturalmente exigem mais processamento que as *tuple-ants*, já que uma *template-ant* precisa executar um grande volume de comparações para encontrar uma tupla que satisfaça as restrições especificas em seu *template*. E isso é computacionalmente dispendioso.

O *nível magnético* a respeito de uma determinada tupla τ_c que algum nó i está exposto é definido pela Equação (5).

$$M_L(i, \tau_c) = Max\{C(n, \tau_c) : n \in NH(i)\} \tag{5}$$

O nível magnético para um nó i corresponde à máxima concentração de tuplas que são similares à tupla τ_c nos nós vizinhos de i. M_L é utilizado para determinar a *força magnética* sofrida pelas formigas presentes no nó i, designada por $F_M(i)$ (Equação (6)). Esta força magnética é responsável por produzir um "comportamento estranho" em uma *tuple-ant*, obrigando ela a depositar sua tupla no nó atual, ou seja, antes do esperado.

$$F_M(i, \tau_c) = M_c(i) * \frac{M_L(i, \tau_c)}{M_c(i) + M_L(i, \tau_c)} \tag{6}$$

A força magnética que afeta as *tuple-ants* no nó i tende à zero quando não existe nenhum campo magnético de nível significante produzido pelos vizinhos do nó

i. $F_M(i, \tau_c)$ depende da constante de *restrição magnética* $M_c(i)$ que representa o número máximo de tuplas que um nó i pode armazenar sem ser considerado "sobrecarregado".

A probabilidade de um "comportamento estranho" durante a fase de movimentação de uma formiga que está atualmente no nó i carregando uma tupla τ_c é definida pela Equação (7).

$$P_S(i, \tau_c) = \frac{F_M(i, \tau_c) + C(i, \tau_c)}{\sum_{\forall n \in NH(i)}(C(n, \tau_c) + Ph(n, \tau_c)) + F_M(i, \tau_c) + C(i, \tau_c)} \quad (7)$$

O "comportamento estranho" de uma *tuple-ant* consiste na decisão de depositar a tupla carregada τ_c no nó corrente ao invés mover-se para outro nó como é esperado. Um nó i que possui um nó adjacente sobrecarregado, por exemplo, está sobre uma forte interferência magnética (calculada a partir de $F_M(i, \tau_c)$) e, consequentemente, uma formiga neste nó possui uma elevada probabilidade de depositar a tupla em i, i.e., antes de chegar ao nó sobrecarregado.

Considerando uma $F_M(i, \tau_c)$ elevada, a influência magnética é potencializada pelo aumento de $C(i, \tau_c)$, que é provocado pela exposição continua do nó ao campo magnético, aumentando a probabilidade de uma formiga carregando uma tupla τ_c apresentar um comportamento estranho em um nó i. Devido a este comportamento, há uma tendência de formação de *clusters* de nós que contém *clusters* de tuplas similares, ou seja, a sobrecarga de nós é evitada através da formação de *clusters* de *clusters*. Observe, no entanto, que a interferência magnética não restringe completamente o comportamento dos indivíduos, pois a adaptabilidade do sistema é alcançada exatamente pela tomada de decisões estocásticas. Portanto, ainda que exista uma intensa interferência magnética, uma formiga pode apresentar um comportamento normal, embora com uma pequena probabilidade.

4. Avaliação Experimental

Com o intuito de avaliar o desempenho do modelo proposto, o *Magnetic SwarmLinda* foi implementado através de simulação. Nos diversos cenários avaliados, que são apresentados na Tabela 1, o desempenho do modelo proposto é comparado aos desempenhos do *SwarmLinda* Tradicional e do *SwarmLinda* com *Anti-Over-Clustering*. A métrica de interesse para a avaliação de desempenho é o tempo médio para a recuperação de uma tupla através da operação `in`. Esta escolha deve-se ao fato da operação de inserção ser instantânea na perspectiva da aplicação, pois não é necessário que o processo cliente aguarde que a tupla seja depositada.

Uma *unidade de tempo ideal* (UTI), representa o tempo necessário para a entrega de uma mensagem entre dois nós diretamente conectados. Foram utilizadas topologias *Small-World* [Watts and Strogatz 1998], no qual inicialmente é criada uma rede em anel com $N = |V|$ nós, sendo que cada nó é conectado com seus k vizinhos mais próximos (se k é par). Em seguida, cada aresta $(u, v) \in E$ é substituída com probabilidade σ por uma nova aresta (u, w), onde $w \in V$ é escolhido aleatoriamente. Foram usados os seguintes parâmetros para a geração de gráficos: a probabilidade $\sigma = 30\%$ de reescrever uma aresta do grafo *Small-World*, o número $N = 16$ de nós do sistema e o grau médio $k = 4$.

Em cada nó que compõe o Espaço de Tuplas é conectado um processo cliente que é responsável por produzir carga sobre o Espaço de Tuplas. Cada cliente executa uma

operação a cada I UTI's, alternando entre operações `out` e `in`. Entretanto, no início da simulação, os processos clientes executam apenas operações `out` até que T tuplas tenham sido depositadas, permitindo que uma carga inicial de tuplas seja estabelecida. Cada operação tem um probabilidade uniforme de ser relativa a qualquer um dos τ^t tipos de tuplas e aos seus τ^v valores possíveis. As operações de recuperação são relativas exclusivamente a tuplas disponíveis no Espaço de Tuplas. Outros parâmetros *default*, que foram variados nos cenários dos experimentos realizados incluem: o número de operações $OP = 1000$ que cada nó é capaz de executar durante uma UTI, a duração $D = 1000$ da simulação em termos de número de operações `in`, o número $S = 50$ de simulações executadas em cada cenário e a taxa de evaporação ρ = 20%.

Além disso, o valor da restrição magnética $M_c(i)$ utilizado nos diferentes cenários foi definido como a quantidade "desejável" de tuplas a ser depositada em cada nó. De tal modo, $M_c(i) = T/N, \forall i \in V$, onde T é o número total de tuplas no sistema (independentemente do seu "tipo") e $N = |V|$ é o número de nós.

Tabela 1. Cenários de Avaliação de Desempenho

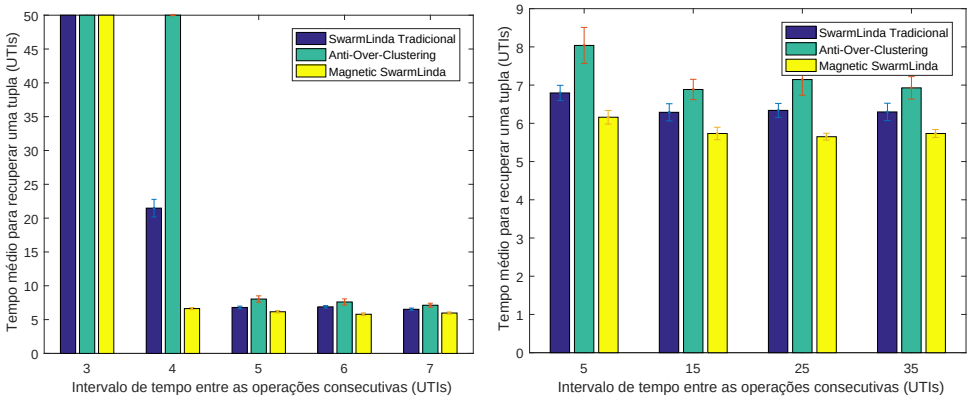
Cenário	Parâmetro	Valores	Avaliação de desempenho para
1	I	3, 4, 5, 6 e 7	altas taxas de requisição de operações
2	I	5, 15, 25 e 35	altas e baixas taxas de requisição de op.
3	τ^v	500, 1000, 1500 e 2000	diferentes valores para cada tipo de tupla
4	k	4, 6, 8 e 10	diferentes números médios de arestas
5	T	(1, 2, 3 e 4) * 10^5	diferentes números de tuplas no sistema
6	τ^t	3, 4, 5, 6, 7 e 8	diferentes números de tipos de tuplas

Nos Cenários 1 e 2, o desempenho é avaliado para diferentes intervalos I entre operações, ou seja, distintas frequências de operações. No Cenário 3, o desempenho é avaliado para diferentes níveis de "raridade" de tuplas, uma vez que o número de tuplas idênticas é maior conforme o número τ^v de valores possíveis de tuplas é menor. No Cenário 4, o desempenho é avaliado quando aumenta a densidade do grafo. No Cenário 5, o desempenho é avaliado para diferentes números T de tuplas depositadas no Espaço de Tuplas, sendo que a quantidade permanece constante já que cada operação de inserção é alternada com uma de recuperação. Finalmente, no Cenário 6, o desempenho é avaliado para diferentes números de tipos de tuplas. Isso permite medir o impacto de diferentes quantidades de *clusters* de nós, uma vez que cada *cluster* - no modelo magnético - está associado a um determinado tipo de tupla. O nível de confiança dos resultados é de 95%.

Os resultados do Cenário 1 apresentados na Figura 3(a) mostram os limites para os níveis de carga (frequência com que operações são executadas) a partir do qual o Espaço de Tuplas deixa de funcionar adequadamente. Este ponto de saturação ocorre quando o tempo médio para recuperar uma tupla é superior ao dobro do intervalo I entre operações, pois o tempo entre as inserções - devido à alternância de operações - é a metade do valor de I. Neste momento, o Espaço de Tuplas não consegue absorver a carga, provocando uma degradação cumulativa do desempenho do sistema.

A identificação do ponto de saturação das três abordagens constata que a estratégia proposta suporta níveis superiores de requisições quando comparada às demais abordagens. O ponto de saturação do *Magnetic SwarmLinda* ocorre para $I = 3$. Para as outras

duas abordagens, este ponto ocorre quando $I = 4$. É importante mencionar que o valor de eixo y foram "aparados" em 50 a fim mostrar melhor o comportamento das três abordagens para $I \geq 4$. Além disso, para a abordagem tradicional, quando $I = 4$, o tempo médio para recuperar uma tupla tende ao infinito para simulações mais longas. Isso acontece porque para $I = 4$, os processos clientes estão inserindo tuplas a cada 8 UTIs, enquanto que o tempo médio para recuperar uma tupla é cerca de 20 UTIs.



(a) Cenário 1 (b) Cenário 2

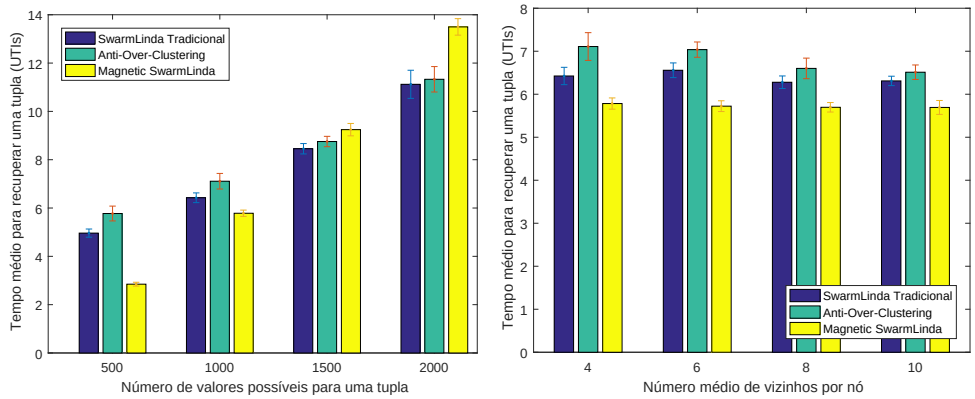
Figura 3. Avaliação de Desempenho dos Cenários 1 e 2

Os resultados do Cenário 2, que são apresentados na Figura 3(b), demonstram que a abordagem proposta apresenta um desempenho superior em momentos nos quais o Espaço de Tuplas está em funcionamento normal, ou seja, não está próximo ao ponto de saturação. Conforme esperado, todas as abordagens têm um desempenho melhor em cenários com baixos níveis de requisições, pois a carga sobre o Espaço de Tuplas é menor.

Os resultados do Cenário 3 na Figura 4(a) demonstram que o modelo proposto apresenta um desempenho melhor quando há uma quantidade menor de possíveis valores por tipo de tupla. Conforme diminui o número de valores possíveis, maior é o número de tuplas idênticas. Desta forma, a quantidade de nós que uma *template-ant* precisa visitar tende a ser menor, uma vez que é maior a probabilidade de existirem tuplas compatíveis em múltiplos nós de um *cluster*. Por outro lado, a abordagem magnética apresenta um desempenho inferior as outras abordagens quando há uma pequena quantidade de tuplas idênticas (ou seja predominam tuplas "raras") pois a *template-ant* é obrigada a visitar um número maior de nós do *cluster* magnético, gerando muito movimento dentro do *cluster*.

Os resultados do Cenário 4 na Figura 4(b) demonstram que a abordagem magnética possui um melhor desempenho no tempo de resposta para recuperação de tuplas para diferentes topologias. Nas topologias com baixa densidade a probabilidade de uma formiga alcançar qualquer *cluster* com um pequeno número de saltos é alta. Como cada *cluster* magnético é composto de múltiplos nós, é provável que exista pelo menos um caminho curto entre qualquer nó do Espaço de Tuplas e um dos nós de cada *cluster*.

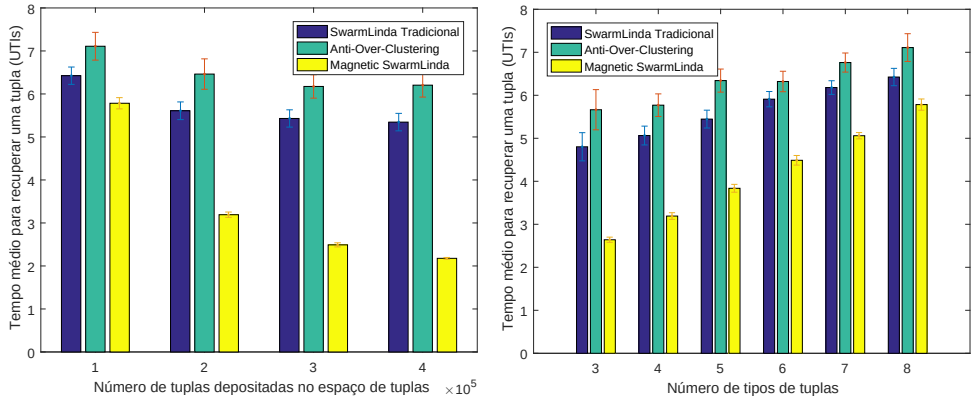
Os resultados do Cenário 5 na Figura 5(a) demonstram que a abordagem proposta possui um desempenho superior às demais abordagens para diferentes quantidades de tuplas depositadas no Espaço de Tuplas. Além disso, o tempo médio para recuperar uma tupla diminui de forma muito expressiva no *Magnetic SwarmLinda* conforme a quantidade de tuplas aumenta. Todas as abordagens apresentam um desempenho melhor para



(a) Cenário 3 (b) Cenário 4

Figura 4. Avaliação de desempenho dos cenários 3 e 4.

grandes quantidades de tuplas, pois há mais tuplas disponíveis no Espaço de Tuplas que são compatíveis com os *templates* das operações de leitura.



(a) Cenário 5 (b) Cenário 6

Figura 5. Avaliação de desempenho dos cenários 5 e 6.

Os resultados do Cenário 6 na Figura 5(b) demonstram que a abordagem proposta é capaz de tirar proveito de cenários onde o número de tipos de tuplas τ^t é menor que o número nós $N = |V|$. Isto deve-se ao fato que conforme diminui o número de tuplas, menor é o número de *clusters*. Desta forma, cada *cluster* é formado por um número maior de nós, provocando uma expressiva melhora do desempenho. As outras abordagens se destacam quando há poucos tipos de tuplas, porém a melhora ocorre de maneira menos expressiva.

5. Conclusão

Espaços de Tuplas são um modelo simples e elegante para a construção de sistemas paralelos e distribuídos. Abordagens distribuídas têm sido propostas para aumentar a escalidade do sistema. Este trabalho apresentou o *Magnetic SwarmLinda* que combina inteligência de enxames e campos magnéticos virtuais para não apenas garantir a escalabilidade mas também permitir a recuperação eficiente de tuplas. Em comparação com outras abordagens, o *Magnetic SwarmLinda* evita a dispersão excessiva de tuplas semelhantes, formando *c*lusters de *c*lusters contendo tuplas similares evita a sobrecarga de nós

específicos. O sistema foi implementado e avaliado em diferentes cenários, demonstrando sua superioridade com relação a outras estratégias exceto em cenários com tuplas "raras" – isto é, pequenas quantidades de tuplas similares – que não são t¿¿picos na computação de alto desempenho. Os benefícios são comprovados sempre sempre que o número de tuplas é elevado. Em particular, houve ganho significativo de desempenho em cenários onde a quantidade de nós que compõem o Espaço de Tuplas é maior que o número de tipos de tuplas. Além disso, o *Magnetic SwarmLinda* apresentou um comportamento estável independentemente da densidade da rede. Trabalhos futuros incluem definir uma abordagem autonômica para ajustar o valor da restrição magnética ao longo do tempo.

Referências

Atkinson, A. (2008). Tupleware: A distributed tuple space for cluster computing. In *Parallel and Distributed Computing, Applications and Technologies, 2008. PDCAT 2008. Ninth International Conference on*, pages 121–126. IEEE.

Banks, A. N. and Srygley, R. B. (2003). Orientation by magnetic field in leaf-cutter ants, atta colombica (hymenoptera: Formicidae). *Ethology*, 109(10):835–846.

Casadei, M., Menezes, R., Tolksdorf, R., and Viroli, M. (2007). On the problem of over-clustering in tuple-based coordination systems. In *Self-Adaptive and Self-Organizing Systems, 2007. SASO'07. First International Conference on*, pages 303–306. IEEE.

de Paula Lima Jr., L. A. and Calsavara, A. (2010). Autonomic application-level message delivery using virtual magnetic fields. *Journal of Network and Systems Management*, 18:97–116.

Gelernter, D. and Bernstein, A. J. (1982). Distributed communication via global buffer. In *Proceedings of the first ACM SIGACT-SIGOPS Symposium on Principles of distributed computing*, pages 10–18. ACM.

Hari, H. (2012). Tuple space in the cloud. Master's thesis, Uppsala University, Department of Information Technology.

Jiang, Y., Xue, G., Jia, Z., and You, J. (2006). Dtuples: A distributed hash table based tuple space service for distributed coordination. In *Grid and Cooperative Computing, 2006. GCC 2006. Fifth International Conference*, pages 101–106. IEEE.

Maia, M. E., Andrade, R., and Viana, W. (2016). Towards a component infrastructure for cyber-physical systems. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 626–628. ACM.

Menezes, R. and Tolksdorf, R. (2003). A new approach to scalable linda-systems based on swarms. In *Proceedings of the 2003 ACM Symposium on Applied computing*, pages 375–379. ACM.

Picco, G. P., Murphy, A. L., and Roman, G.-C. (1999). Lime: Linda meets mobility. In *Proceedings of the 21st international conference on Software engineering*, pages 368–377. ACM.

Rowstron, A. (1998). Wcl: A co-ordination language for geographically distributed agents. *World Wide Web*, 1(3):167–179.

Wajnberg, E., Acosta-Avalos, D., Alves, O. C., de Oliveira, J. F., Srygley, R. B., and Esquivel, D. M. (2010). Magnetoreception in eusocial insects: an update. *Journal of the Royal Society Interface*, page rsif20090526.

Waldo, J. et al. (1997). Javaspace specification-revision 0.4. Technical report, Technical report, Sun Microsystems, JavaSoft Lab.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of "small-world" networks. *nature*, 393(6684):440–442.

Um Modelo Analítico para Estimar o Consumo de Energia de Aplicações Web no Nível de Transações

Alex R. Ferreira[1], Denner S. L. Vidal[2], Valéria Q. dos Reis[2], Sand Luz Correa[1]

[1]Instituto de Informática – Universidade Federal de Goiás

[2]Faculdade de Computação – Universidade Federal de Mato Grosso do Sul

alexrabeloferreira@inf.ufg.br, denner.vidal@gmail.com

valeria@facom.ufms.br, sand@inf.ufg.br

Abstract. *Modern enterprise servers provide several features to manage the power consumption of their subsystems. In environments with high computational demand, these features can significantly reduce energy consumption. However, in order to make proper use of such resources, we need to understand how the applications that run in these servers consume the computational resources and the implications of such usage for the overall power and energy consumption. In this work, we propose an energy model to characterize the energy consumption of enterprise servers that host Web systems. Unlike other works in the literature, our model captures the consumption pattern of these applications at the level of transactions and for each tier of the system. In addition, our model depends solely on the CPU utilization and on server architectural parameters, which can be easily obtained in current production environments. We demonstrate the effectiveness of our model in a real experimental environment, based on the TPC-W benchmark. Results show that our model is able to estimate the energy consumption of Web applications with errors in the same order of magnitude as those presented by previous work.*

Resumo. *Servidores modernos oferecem diversos recursos para gerenciar o consumo de potência de seus subsistemas. Em ambientes com grande demanda computacional, esses recursos podem reduzir o consumo de energia de maneira significativa. No entanto, para fazer bom uso de tais recursos, é necessário entender como as aplicações que executam nesses servidores utilizam os recursos computacionais e as implicações desse uso para o consumo de potência e de energia. Neste trabalho, propomos um modelo analítico para estimar o consumo de energia de servidores que hospedam sistemas Web. Diferentemente de outros trabalhos, nosso modelo captura o padrão de consumo desses sistemas na granularidade de transações e para cada camada do sistema. Além disso, nosso modelo se baseia apenas na utilização de CPU e em parâmetros arquiteturais do servidor, os quais podem ser facilmente obtidos nos ambientes de produção atuais. Demonstramos a efetividade do nosso modelo em um ambiente de experimentação real, baseado no benchmark TPC-W. Resultados mostram que nosso modelo é capaz de estimar o consumo de energia de sistemas Web com erros na mesma ordem de grandeza que modelos mais complexos existentes na literatura.*

1. Introdução

O custo para alimentar um servidor empresarial típico durante seu tempo de vida já ultrapassa o custo de aquisição [Scaramella et al. 2014]. Como consequência, nos últimos anos, ambientes reconhecidamente orientados a desempenho, como *data centers* para computação em nuvem e *clusters* para processamento de alto desempenho (*High Performance Computing - HPC*), passaram a reconhecer a eficiência energética como outro requisito importante no projeto de sistemas computacionais de grande escala [Barroso and Hölzle 2007].

Servidores modernos oferecem diversos recursos para gerenciar o consumo de potência de seus subsistemas. Para ilustrar, a partir da tecnologia Sandy Bridge [Rotem et al. 2012], os processadores Intel proveem uma interface onde é possível limitar o consumo de potência em um determinado limiar, durante um período de tempo. Muitos processadores permitem também ajustar a voltagem e a frequência de sua operação de acordo com a carga de trabalho ou nível de utilização [Hsu and Feng 2005, Rountree et al. 2009], enquanto outros *chips* podem desativar ou colocar em estágio de dormência um subcomponente que não está em uso [Kaxiras and Martonosi 2008].

Em ambientes com grande demanda computacional, a aplicação dessas técnicas juntamente com controles no nível do sistema operacional, como por exemplo alocação de *threads* e políticas de escalonamento, podem reduzir o consumo de energia de maneira significativa [Isci et al. 2006a, Fan et al. 2007, Cochran et al. 2011]. No entanto, para fazer bom uso de tais técnicas, é necessário entender como as aplicações que executam nesses servidores utilizam os recursos computacionais disponíveis e as implicações desse uso para o consumo de potência e, por conseguinte, de energia. Além disso, como mostrado em [Isci et al. 2006b], quanto mais fino for o entendimento e o controle sobre os componentes controlados, maior a economia de energia. Esse controle fino se torna ainda mais importante em ambientes como *data centers* baseados em serviços de Internet, onde fenômenos de rajada e atrasos entre as diferentes camadas que compõem os serviços podem alterar o padrão de utilização dos recursos de maneira considerável e em um curto espaço de tempo [Mi et al. 2010]. A despeito desse fato, poucos trabalhos em eficiência energética em sistemas ou serviços de Internet têm como foco a caracterização do consumo de energia em um nível mais detalhado. Como consequência, a fim de obter maior acurácia, muitos modelos existentes na literatura, como em [Piga et al. 2014], procuram monitorar um grande número de métricas e contadores de desempenho, para então realizar algum mecanismo de poda que retorne as métricas mais relacionadas com o componente observado. Essa abordagem, no entanto, torna os modelos de energia mais complexos, além de gerar sobrecarga relacionada à coleta e armazenamento dos dados coletados.

Neste trabalho, tratamos esse problema propondo um modelo analítico que captura o padrão de consumo de energia de sistemas Web na granularidade de páginas Web e para cada camada do sistema. Além disso, nosso modelo se baseia apenas na utilização de CPU e em parâmetros arquiteturais do servidor, os quais podem ser facilmente obtidos nos ambientes de produção atuais. Nosso modelo se baseia em um arcabouço proposto por [Zhang et al. 2007], o qual permite estimar demandas de CPU geradas pelo processamento de páginas Web em um hardware particular. Usamos então essas demandas para parametrizar um modelo tradicional de consumo de energia que, juntamente com um método de regressão linear, nos permite estimar o custo energético necessário para pro-

cessar cada página individualmente. Avaliamos a efetividade do nosso modelo utilizando um ambiente de experimentação real e o *benchmark* TPC-W [TPC 2016]. Os resultados obtidos mostram que nosso modelo é capaz de estimar o consumo de energia para um sistema Web com um erro relativo médio de 6,5% para o pior cenário, enquanto modelos mais complexos da literatura apresentam erros com a mesma ordem de grandeza.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 introduz uma breve revisão de trabalhos relacionados. A Seção 3 apresenta o modelo de sistema considerado e descreve um modelo da literatura para estimar demandas de CPU em sistemas Web. A Seção 4 descreve o modelo de estimação de consumo de energia proposto. O ambiente de experimentação bem como o *benchmark* usado na avaliação são apresentados na Seção 5 juntamente com os resultados obtidos. Finalmente, a Seção 6 conclui o trabalho e apresenta as perspectivas de trabalhos futuros.

2. Fundamentos e Trabalhos Relacionados

Potência e energia são comumente definidas em termos do trabalho realizado por um sistema. Energia é a quantidade total de trabalho realizada durante um período de tempo, enquanto potência é a taxa com a qual o sistema executa o trabalho. Em termos formais:

$$E = T \times P, \tag{1}$$

onde E denota a energia, P denota a potência e T representa um intervalo de tempo específico. Em circuitos CMOS, o consumo de potência é proveniente de duas fontes: uma estática (P_{static}) e outra dinâmica($P_{dynamic}$) [Kaxiras and Martonosi 2008]. Logo, podemos reescrever a Equação 1 como:

$$E = T \times (P_{static} + P_{dynamic}). \tag{2}$$

O consumo estático é causado por "vazamentos" de correntes presentes em qualquer circuito ativo. Esse tipo de consumo não depende do cenário de utilização e está presente mesmo quando o componente está ocioso. O consumo dinâmico, por outro lado, é proveniente de atividades nos circuitos elétricos e está relacionado com o cenário de utilização, frequências do relógio e atividades de entrada e saída.

Em processadores, o problema de estimação de potência é definido como a tarefa de mensurar ou aferir a potência consumida por esses componentes quando um programa ou carga de trabalho são executados. Como a maioria dos processos de medição real para esse problema são intrusivos [Bedard et al. 2010, Ge et al. 2010], muitos trabalhos utilizam modelos de estimação em software. Nesse contexto, uma formulação comumente empregada consiste em estimar P_{static} através de parâmetros arquiteturais, enquanto $P_{dynamic}$ é estimada como uma função linear de n registradores específicos, denominados contadores de desempenho, que armazenam estatísticas sobre a atividade de diferentes subsistemas do processador [Joseph and Martonosi 2001, Isci and Martonosi 2003]. A potência total consumida é então calculada como a soma desses dois valores. No entanto, a maioria dos trabalhos que se baseiam em contadores de desempenho para estimar o consumo de potência/energia de processadores são voltados para ambientes de HPC, cujas cargas de trabalho diferem significativamente daquelas encontradas em servidores Web [Wang et al. 2013]. Além disso, a maioria desses modelos requerem mais de 15 contadores de desempenho, alguns dos quais não estão disponíveis na maioria dos servidores

comerciais típicos. Nosso modelo, ao contrário, baseia-se apenas na utilização de CPU e em parâmetros arquiteturais do servidor.

A proposta de [Bohrer et al. 2002] foi a primeira a tratar do problema de estimação de consumo de potência/energia em servidores e sistemas Web. Os autores apresentam uma ferramenta de simulação que estima o tempo de CPU e a energia consumida por uma requisição Web com base nos ciclos de CPU gastos para executar a requisição. Entretanto, o modelo de estimação leva em consideração apenas o servidor de aplicação e os autores assumem que as requisições ou são completamente servidas pela memória ou são completamente atendidas pelo disco. Nosso modelo, ao contrário, leva em consideração todos os objetos recuperados nas diferentes camadas (lógica e de dados) para atender as requisições. Consequentemente, nosso modelo é capaz de caracterizar a demanda de CPU e o consumo de energia requerido por cada página Web em cada camada.

Uma proposta para estimar o consumo de potência/energia de sistemas Web em diferentes estados de frequência/voltagem (*P-state*) de um processador é apresentado em [Piga et al. 2014]. Para derivar a relação entre consumo de potência e *P-state*, os autores monitoram a atividade do sistema através de um grande número de contadores de desempenho e métricas do sistema operacional. Entretanto, como o monitoramento dessas métricas é feito como uma caixa preta, ou seja, sem associá-las às requisições que estão sendo atendidas, para alcançar maior acurácia, os autores utilizam diferentes métodos de análise de dados, incluindo regressão linear, análise de *cluster* e técnicas de inteligência artificial. Dessa forma, os modelos apresentados atingem erros médios próximos a 2%. Como mostraremos na Seção 5.2, nosso modelo, apesar de usar apenas a técnica de regressão linear e um número menor de métricas, é capaz de atingir erros médios na mesma ordem de grandeza.

Existem também trabalhos que investigam os efeitos das técnicas de gerenciamento de energia no desempenho de servidores Web [Wang et al. 2013, Metri et al. 2012]. Esses trabalhos, entretanto, não proveem nenhuma formulação para o problema de estimação de consumo de potência/energia. Finalmente, existe uma vasta literatura em simulação que busca melhorar a eficiência energética em *data centers* de computação em nuvem pela consolidação de máquinas virtuais (VMs)(um *survey* abrangente pode ser encontrado em [Beloglazov and Buyya 2012]). Nosso modelo pode ser usado nesses simuladores para guiar a decisão dos algoritmos de otimização.

3. Demanda de CPU Requerida por uma Página Web

Tipicamente, sistemas Web são concebidos em múltiplas camadas. Cada camada, por sua vez, fornece um certa funcionalidade, incluindo apresentação de dados, lógica da aplicação e gerenciamento de dados. Clientes interagem com esses sistemas através de navegadores, os quais recuperam páginas Web. Uma página Web consiste em um arquivo HTML e vários objetos embutidos (imagens, áudio, vídeo, etc.). No lado do servidor, a recuperação de uma página Web envolve o processamento de vários objetos menores. Se a recuperação requer acesso a dados, o servidor de aplicação encaminha a solicitação para a camada de dados, a qual executa um servidor de banco de dados. Referimo-nos à combinação de todas as atividades de processamento no lado do servidor para entregar uma página Web para um cliente como uma *transação*. Isto inclui as operações para gerar o arquivo HTML principal, bem como para recuperar os objetos embutidos e executar

consultas de banco de dados.

Zhang et al. [Zhang et al. 2007] propuseram um modelo analítico simples e acurado para modelar cargas Web usando informações no nível de transações. A ideia básica consiste em usar o conhecimento da demanda de CPU de cada tipo de página Web para então compor a demanda de CPU de cargas de trabalho contendo diferentes combinações de páginas. Assumindo um sistema Web de n camadas e composto por η tipos de páginas Web diferentes, os autores aplicam a Lei da Utilização [Menasce et al. 2004], obtendo:

$$\sum_{i=1}^{\eta} \eta_i \times D_{i,j} = U_{CPU,j} \times T, \tag{3}$$

onde T é o tamanho da janela de monitoramento; η_i, $1 \leq i \leq \eta$, é o número de páginas Web do tipo i observadas durante o período T; $U_{CPU,j}$, $1 \leq j \leq n$, é a utilização média de CPU na camada j durante o período de tempo T; e $D_{i,j}$ é o tempo médio de serviço de CPU requerido para processar uma página Web do tipo i na camada j. Entretanto, como é difícil obter tempos de serviço de forma acurada, os autores propõem uma aproximação e substituem $D_{i,j}$ pelo custo de CPU (em termos de utilização) de $D_{i,j}$, denotado por $C_{i,j}$. Assim, para cada camada j, uma utilização média aproximada, denotada por $U'_{CPU,j}$, pode ser calculada como:

$$U'_{CPU,j} = \frac{\sum_{i=1}^{\eta} \eta_i \times C_{i,j}}{T}. \tag{4}$$

Portanto, se observarmos as páginas Web processadas durante janelas de monitoramento T, bem como a utilização $U_{CPU,j}$ em cada camada do sistema durante essas janelas, podemos aproximar $U_{CPU,j}$ de $U'_{CPU,j}$ e empregar uma técnica de regressão linear para estimar os valores $C_{i,j}$. Esses valores representam a utilização média de CPU requerida por cada tipo de página Web em cada camada do sistema. Na seção a seguir, mostraremos como usamos essa informação para estimar o consumo de energia requerido por cada tipo de página em cada camada de um sistema Web.

4. Demanda de Energia Requerida por uma Página Web

Nossa proposta leva em consideração apenas a potência/energia consumida pelo processador, uma vez que esse componente é o responsável pela maior parte da energia consumida nos servidores típicos [Beloglazov and Buyya 2012]. Para estimar a potência do processador, usamos uma abordagem tradicionalmente empregada [Isci and Martonosi 2003], a qual consiste em aproximar $P_{dynamic}$ de um fator α da potência ativa. Essa última, denotada por P_{active}, representa a diferença entre a potência máxima que o processador pode lidar (*thermal design power*), representado por P_{TDP}, e P_{static} ou seja:

$$P_{dynamic} \approx \alpha \times P_{active}. \tag{5}$$

$$P_{active} = P_{TDP} - P_{static}. \tag{6}$$

Tipicamente, os parâmetros P_{TDP} e P_{static} podem ser facilmente obtidos a partir de especificações do hardware. Para descrever o fator de atividade α, usamos a utilização média de CPU durante um intervalo de tempo T, normalizado por um fator que representa a utilização máxima do processador, ou seja:

$$\alpha = \frac{U_{CPU}}{MAX_{U_{CPU}}}. \tag{7}$$

Em nosso modelo de sistema, assumimos que os servidores são equipados com processadores com múltiplos núcleos. Seja U_{core_k} a utilização média do núcleo k durante a janela de tempo T. Modelamos a utilização média de um processador com m núcleos no intervalo T como sendo a soma da utilização média de cada núcleo durante T, isto é:

$$U_{CPU} = \sum_{k=1}^{m} U_{core_k}. \tag{8}$$

Modelamos a utilização máxima de um processador ($MAX_{U_{CPU}}$) como o produto do número de núcleos (m) e a utilização máxima de cada núcleo (100). Portanto, usando as Equações 2, 5 e 7, modelamos o consumo total de energia de um servidor com um processador com m núcleos como:

$$E = T \times (P_{static} + \frac{\sum_{k=1}^{m} U_{core_k}}{100m} \times P_{active}). \tag{9}$$

Por simplicidade, assumimos que um sistema Web com n camadas e formado por η tipos de páginas Web é implantado em um único servidor físico equipado com m núcleos de processamento. Assumimos também que cada camada j do sistema é implantada em uma VM diferente e cada VM é instanciada com um número fixo m_j de núcleos. Para capturar o padrão de consumo dos recursos computacionais ao longo do tempo, observamos o sistema durante janelas de monitoramento de tamanho T. Ao final de cada janela de monitoramento, gravamos as seguintes informações:

- η_i: o número de páginas Web do tipo i processadas durante a janela de monitoramento, $1 \leq i \leq \eta$;
- $U_{CPU,j}$: a soma da utilização média de CPU de todos os núcleos de uma camada j durante a janela de monitoramento, $1 \leq j \leq n$;

Denotando as potências estática e ativa do servidor por P_{static} e P_{active}, respectivamente, e abstraindo a camada j como um servidor com m_j núcleos, podemos aplicar a Equação 9 e calcular a energia consumida pela camada j em uma janela de tempo T como:

$$E_j = T \times (P_{static,j} + \frac{\sum_{k_j=1}^{m_j} U_{core_{k_j}}}{100m_j} \times P_{active,j}), \tag{10}$$

onde:

$$P_{static,j} = m_j \times \frac{P_{static}}{m}, \qquad P_{active,j} = m_j \times \frac{P_{active}}{m} \quad \text{e} \quad \sum_{k_j=1}^{m_j} U_{core_{k_j}} = U_{CPU,j}.$$

Sejam $C_{i,j}$ e $T_{i,j}$, respectivamente, a utilização de CPU e o tempo médio necessários para processar uma página Web do tipo i na camada j. Assumindo que toda atividade realizada nessa camada é decorrência apenas do processamento de transações, podemos calcular a energia requerida para processar uma página Web desse tipo nessa camada como:

$$E_{\tau_i,j} = T_{i,j} \times \frac{C_{i,j}}{100m_j} \times P_{active,j}, \tag{11}$$

Como dentro de uma janela de monitoramento T o servidor processa η_i páginas Web do tipo i, a energia consumida por esse tipo de página na camada j durante T é $\eta_i \times E_{\tau_i,j}$. Logo, a soma da energia consumida por todas as páginas de todos os tipos na camada j durante T resulta na energia proveniente do consumo dinâmico dessa camada. Dessa forma, podemos reescrever a Equação 10 como:

$$E_j = T \times P_{static,j} + \sum_{i=1}^{\eta} \eta_i \times E_{\tau_i,j}. \tag{12}$$

Combinando as Equações 11 e 12, obtemos a Equação 13 para cada janela de monitoramento.

$$E_j = T \times P_{static,j} + \sum_{i=1}^{\eta} (\eta_i \times T_{i,j} \times \frac{C_{i,j}}{100 m_j} \times P_{active,j}). \tag{13}$$

Na Equação 13, os parâmetros $P_{static,j}$ e $P_{active,j}$ são calculados a partir dos parâmetros arquiteturais P_{static} e P_{active}. Como $U_{CPU,j}$ é gravado no final de cada janela de monitoramento, E_j pode ser estimado facilmente usando a Equação 10. Como η_i também é gravado no final de cada janela de monitoramento, $T_{i,j}$ e $C_{i,j}$ são as variáveis desconhecidas. Contudo, como gravamos a soma da utilização média de CPU de todos os núcleos de uma camada j ao final de cada janela de monitoramento ($U_{CPU,j}$), podemos facilmente calcular $C_{i,j}$ utilizando o arcabouço descrito na Seção 3. Uma vez estimados os valores $C_{i,j}$, podemos substituí-los na Equação 13, para então aplicar um método de regressão linear e calcular os valores $T_{i,j}$. Finalmente, substituindo $C_{i,j}$ e $T_{i,j}$ por seus respectivos valores na Equação 11, podemos estimar a energia requerida por cada tipo de página Web em cada camada do sistema.

5. Avaliação de Desempenho

Para realizar a avaliação do modelo proposto, preparamos um ambiente de testes com um sistema *Web* multicamadas, descrito na Seção 5.1. Os resultados obtidos em experimentos que comparam os valores estimados pelo modelo de energia proposto e os valores retornados por um modelo implementado em hardware são apresentados na Seção 5.2.

5.1. Infraestrutura de Testes e Carga de Trabalho

Nosso ambiente de experimentação é baseado no TPC-W [TPC 2016], um *benchmark* que emula um *site* típico de comércio eletrônico seguindo uma arquitetura em três camadas. A Figura 1 ilustra o ambiente implantado, o qual inclui uma camada de aplicação (`APP VM`) baseada no servidor Apache Tomcat, uma camada de dados (`DB VM`) implementada pelo servidor MySQL e uma camada de apresentação (`Emulator VM`) contendo o módulo TPC-W que emula os clientes do *site*. Cada camada é implementada em uma VM diferente. As camadas de aplicação e dados foram hospedadas no mesmo servidor físico (`Servidor 1`), o qual possui um processador Intel Xeon E5-2620 v3 2.4 GHz com 16 GB de memória RAM. A camada de apresentação foi implantada em outro servidor (`Servidor 2`), equipado com um processador Intel Xeon E5-2420 v2 2.2 GHz com 32GB de memória RAM.

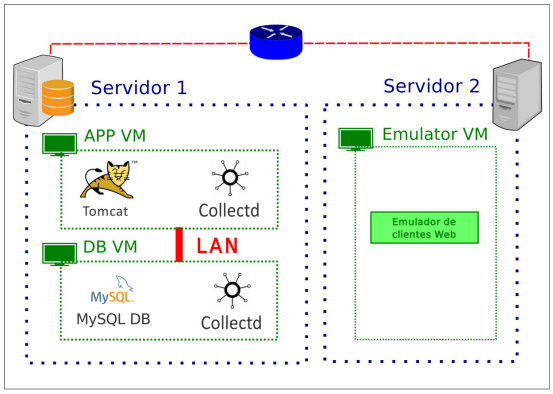


Figura 1. Arquitetura do ambiente de testes.

VM	Amazon EC2	VCPU	RAM
`APP`	t2.medium	2	4GB
`DB`	t2.medium	2	4GB
`Emulator`	t2.small	1	2GB

Tabela 1. Configurações das máquinas virtuais.

Neste trabalho, utilizamos uma implementação em Java de código aberto do TPC-W. Todas as VMs e servidores executam o sistema operacional Linux com a distribuição Ubuntu 14.04. O *hypervisor* instalado nos servidores é o Xen 4.4.1. As máquinas virtuais (`APP VM e DB VM`) também executam a ferramenta collectd [collectd 2016] para realizar as coletas de dados. A Tabela 1 ilustra as principais configurações das máquinas virtuais empregadas. Para melhor emular o comportamento de um sistema real, elas seguem configurações de instâncias Amazon EC2 [1].

Como mostrado na Tabela 2, o TPC-W possui 14 páginas Web, classificadas como requisições relacionadas com a navegação de clientes no *site* ou como requisições de compras de produtos. Essas páginas Web são combinadas para compor três tipos de carga de trabalho, a saber: *Browsing* (95% de navegação e 5% de compra), *Shopping* (80% de navegação e 20% de compra) e *Ordering* (50% de navegação e 50% de compra).

Navegação	Compra
Home	Shopping Cart
New Products	Customer Registration
Best Sellers	Buy Request
Product detail	Buy Confirm
Search Request	Order Inquiry
Search Results	Order Display
	Admin Request
	Admin Confirm

Tabela 2. Páginas do TPC-W de acordo com cada tipo de requisição.

Tipicamente, cargas Web não são limitadas por CPU, uma vez que sistemas Web tendem a executar um grande número de operações de entrada/saída [Wang et al. 2013]. Em geral, a maioria dessas operações ocorrem na camada de dados e, portanto, essa tende a ser a camada mais requisitada. A carga *Browsing* simula um cenário onde existem poucos compradores e várias requisições de busca ou pesquisa. Por outro lado, a carga *Ordering* simula um cenário com um número de pedidos e compras significantemente

[1] https://aws.amazon.com/pt/ec2/instance-types/

maior que a *Browsing*. *Shopping*, por sua vez, é uma carga de trabalho intermediária em relação às duas cargas anteriores.

O TPC-W simula sessões de clientes Web através de emuladores denominados *Remote Browser Emulators* (RBEs). Durante um experimento, o número de RBEs que executam de forma concorrente é mantido constante. Além disso, para cada RBE, o TPC-W define, de forma estatística, o tamanho da sessão (em média 15 minutos), o tempo entre ações do usuário (*think time*, em média 7 segundos) e as consultas que o RBE realiza.

Em nossos experimentos, utilizamos os três tipos de carga do TPC-W. Para emular intensidades de cargas diferentes, executamos um conjunto de experimentos variando o número de RBEs concorrentes em 100, 200, 300, 400 e 500. Executamos cada experimento por 5 horas e 40 minutos, uma vez que esse tempo de vida se mostrou adequado para gerar dados suficientes para o uso da técnica de regressão. Os primeiros e últimos 20 minutos de cada experimento foram considerados períodos de aquecimento e resfriamento, respectivamente. Portanto, esses períodos foram omitidos de nossa análise. As Figuras 2 e 3 ilustram a média de utilização de CPU para cada valor de RBE e para as camadas de aplicação e dados, respectivamente. Nesse experimento, os dados foram coletados a cada 1 minuto. Conforme esperado, a utilização de CPU é menor para a camada de aplicação. Na camada de dados, o uso de CPU é maior e apresenta maior variação. Podemos notar que quanto maior for o número de requisições do tipo `Compra` em uma carga, maior a demanda por CPU na camada de dados. Assim, nessa camada, a carga *Browsing* requer uma demanda por CPU menor que a *Shopping* que, por sua vez, requer uma demanda menor que a *Ordering*.

A carga *Ordering*, diferentemente das demais, apresenta uma curva de utilização média de CPU que culmina em 300 RBEs. A partir desse número de emuladores, o número de itens do banco de dados (10.000) não é suficiente para atender as requisições emitidas, forçando o servidor Web a abortar as requisições. Por esse motivo, a carga *Ordering* utilizada tem limite máximo de 300 RBEs. Para as cargas *Browsing* e *Shopping* observamos comportamento semelhante para RBEs maiores que 500.

A Figura 4 ilustra a energia média consumida pelo processador do `Servidor 1` durante a execução dos experimentos envolvendo os três tipos de cargas. Esse dado foi coletado usando a interface RAPL [David et al. 2010], um modelo em hardware de estimação de potência para todo o chip do processador e disponível nos processadores Intel mais recentes. Nesses experimentos, programamos a interface para realizar as coletas a cada 15 segundos. Esse valor menor foi escolhido para evitar a sobrescrita dos contadores usados pela interface. Para obter a medição de energia dentro de uma janela de 1 minuto, agregamos 4 janelas consecutivas. Podemos notar que as curvas de energia apresentam comportamento similar às curvas de CPU da camada de dados. Esse comportamento era esperado, uma vez que o consumo de CPU na camada de aplicação é pequeno. Podemos notar também que, embora as curvas de CPU (camada de dados) e energia sigam a mesma tendência, a variação na curva de energia é menor. Isso é explicado pelo consumo de potência estática, o qual está presente independentemente da carga executada.

Durante a execução dos experimentos, coletamos e gravamos os valores $U_{CPU,j}$ e η_i a cada minuto. O primeiro valor foi obtido a partir da ferramenta `collectd` instalada em cada VM, enquanto o segundo foi extraído das requisições HTTP registradas

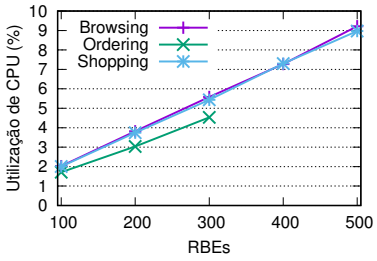


Figura 2. Utilização de CPU para a camada de aplicação.

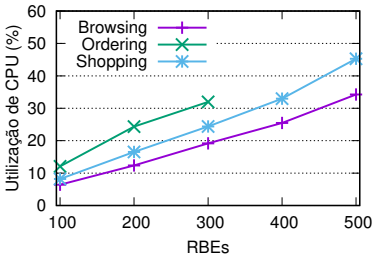


Figura 3. Utilização de CPU para a camada de dados.

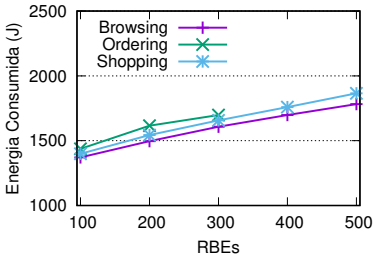


Figura 4. Consumo de energia durante os experimentos.

no arquivo de *log* do Tomcat. Coletamos também a soma da utilização média de CPU de todos os núcleos do `Servidor 1` que não estão sendo usados por nenhuma camada do sistema ($U_{CPU,0}$), bem como o consumo médio de energia pelo processador durante a janela de monitoramento (E). Embora esses dois últimos valores não são usados pelo modelo, eles foram necessários para sua validação. Durante os experimentos, foi utilizado o algoritmo de regressão linear múltipla disponível na ferramenta de computação estatística `R`. Inicialmente, cada carga de trabalho foi executada duas vezes, resultando em 10 horas de execução para cada carga após a remoção do período de aquecimento e resfriamento. Os dados gerados pela primeira execução foram usados para treinar o modelo, enquanto os provenientes da segunda execução foram usados para o teste da regressão.

5.2. Resultados

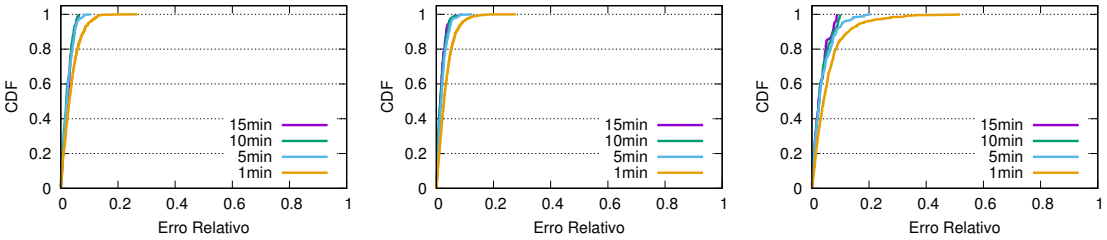
Nas figuras a seguir, os resultados dos experimentos são avaliados para diferentes tamanhos de janelas de monitoramento. Os valores das janelas bem como dos demais parâmetros usados no modelo estão ilustrados na Tabela 3.

Variável	T (mins)	m	n	η	m_j	m_0	P_{TDP} (Watt)	P_{static} (Watt)
Valor	1, 5, 10, 15	6	2	14	2	2	85	19,2

Tabela 3. Valores dos parâmetros utilizados nos experimentos.

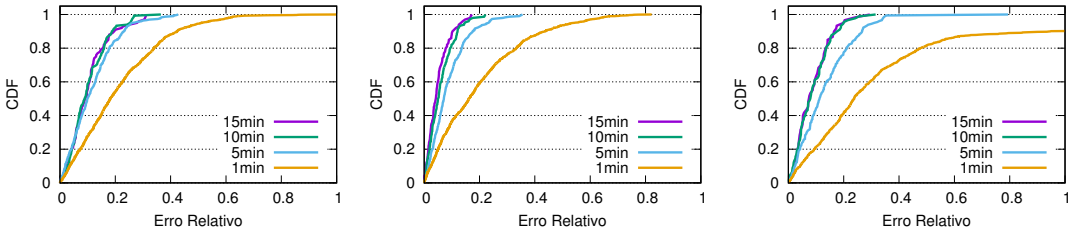
Inicialmente, avaliamos a acurácia do modelo para estimar a demanda de CPU para cada página Web em uma determinada camada, ou seja, os valores $C_{i,j}$. Para isso, utilizamos o erro relativo entre os valores $U'_{CPU,j}$ e $U_{CPU,j}$. O primeiro valor representa a utilização média prevista pelo modelo para uma camada j do sistema a partir dos valores estimados para $C_{i,j}$. O segundo valor representa a utilização média de CPU que de fato foi observada na camada j durante os experimentos. O erro relativo é então obtido da seguinte forma: $\epsilon_{CPU} = \frac{|U'_{CPU,j} - U_{CPU,j}|}{U_{CPU,j}}$.

As Figuras 5 e 6 ilustram as funções de distribuição acumulada (CDFs) de ϵ_{CPU} nas camadas de aplicação e de dados, respectivamente, para tamanhos diferentes da janela de monitoramento. Primeiramente, observamos que os resultados são melhores para a camada de aplicação, onde os valores médio de ϵ_{CPU} para os melhores cenários (janela de 15 minutos) são aproximadamente 2%, 1%, 3% para as cargas *Browsing*, *Shopping* e *Ordering*, respectivamente. Isso ocorre porque nessa camada a utilização de CPU não tem grandes variações. Observamos também que, em geral, janelas de monitoramento



(a) Browsing (b) Shopping (c) Ordering

Figura 5. CDFs de ϵ_{CPU} para a camada de aplicação (APP VM).



(a) Browsing (b) Shopping (c) Ordering

Figura 6. CDFs de ϵ_{CPU} para a camada de dados (DB VM).

maiores resultam em erros menores, especialmente na camada de dados. Isso acontece porque quanto maior uma janela de monitoramento, maior o número de amostras usadas no cálculo da média de CPU e, consequentemente, menor a interferência que amostras de CPU com grandes variações têm sobre a média final. Para a camada de dados, os valores médios de ϵ_{CPU} nos melhores cenários (janela de 15 minutos) são aproximadamente 10%, 5%, 8% para as cargas *Browsing*, *Shopping* e *Ordering*, respectivamente.

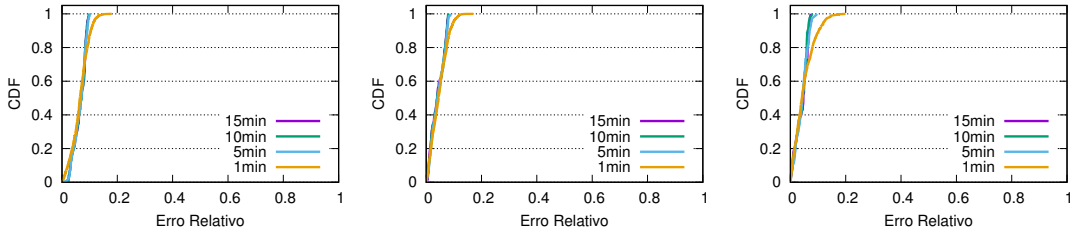
Em seguida, avaliamos a acurácia do modelo para estimar a demanda de energia para cada página Web em uma determinada camada, ou seja, os valores $T_{i,j}$. Como o RAPL não é capaz de reportar o consumo de energia por núcleos do processador, nossa avaliação leva em consideração a energia consumida por todo o processador. O erro relativo é então calculado considerando-se E e E'. O primeiro valor representa a energia consumida pelo servidor em uma janela de monitoramento, sendo esse valor reportado pelo RAPL. O segundo valor representa a energia estimada para o servidor a partir dos valores estimados para E_j usando nosso modelo, bem como a energia consumida pelos núcleos que não estão sendo usados por nenhuma camada do sistema (E_0). Formalmente temos: $E' = E_0 + \sum_{j=1}^{n} E_j$. Assumindo que existam m_0 núcleos do servidor que não estão sendo usados por nenhuma camada no sistema, E_0 é dado por:

$$E_0 = T(P_{static,0} + \frac{\sum_{k_0=1}^{m_0} U_{core_{k_0}} P_{active,0}}{100 m_0}), \tag{14}$$

onde:

$$P_{static,0} = m_0 \frac{P_{static}}{m}, \qquad P_{active,0} = m_0 \frac{P_{active}}{m} \quad \text{e} \quad \sum_{k_0=1}^{m_0} U_{core_{k_0}} = U_{CPU,0}.$$

O erro relativo para a regressão de energia é então obtido da seguinte forma: $\epsilon_{energy} = \frac{|E'-E|}{E}$.



(a) Browsing (b) Shopping (c) Ordering

Figura 7. CDFs de ϵ_{energy} para diferentes cargas.

T (mins)	Browsing	Shopping	Ordering
1	0,065	0,046	0,049
5	0,065	0,042	0,042
10	0,065	0,041	0,040
15	0,065	0,041	0,041

Tabela 4. Valor médio de ϵ_{energy} em cada cenário avaliado.

A Figura 7 ilustra as CDFs de ϵ_{energy} para as cargas *Browsing*, *Ordering* e *Shopping* em diferentes tamanhos de janelas de monitoramento. A Tabela 4 mostra o valor médio de ϵ_{energy} em cada cenário ilustrado na Figura 7. Assim como na regressão de $C_{i,j}$, observamos que janelas de monitoramento de 1 minuto produzem resultados menos precisos que as demais janelas. Notamos também que a acurácia do modelo de energia é, em geral, melhor que a do modelo de estimação de CPU, especialmente quando comparada com o desempenho desse último na camada de dados. Isso ocorre por dois fatores: i) A validação do modelo de energia está levando em consideração todo o servidor e, consequentemente, o desempenho menos acurado na camada de dados está sendo amortizado pelo bom desempenho na camada de aplicação; ii) A pequena variação observada para os valores de energia à medida que o número de RBEs concorrentes aumenta. Como explicado anteriormente, essa pequena variação ocorre devido à componente estática do consumo de potência, a qual está presente independentemente da carga utilizada. Finalmente, observamos que nosso modelo, apesar de utilizar poucos parâmetro e informações facilmente obtidas em ambientes de produção, produz erros médios menores que 7% em qualquer cenário de utilização. Esses erros estão na mesma ordem de grandeza dos apresentados em [Piga et al. 2014].

6. Conclusão

Este trabalho apresentou a proposta e avaliação de um modelo analítico para estimar o consumo de energia de sistemas Web no nível de transações. Utilizando o *benchmark* TPC-W, conseguimos mostrar através de experimentos reais, que o entendimento e o controle mais finos dos componentes do sistema ajudam a construir modelos simples e acurados para estimação de consumo de energia de ambientes com grande demanda computacional. Baseando-se em tais informações, nosso modelo foi capaz de estimar o consumo de energia com erros variando entre entre 4% e $6,5\%$. Esses valores são compatíveis com o estado da arte na área de eficiência energética em *data centers*. Nosso método de estimação de energia diferencia-se por: i) modelar o comportamento de transações Web em vez de sessões; ii) requerer poucas métricas, sendo essas de fácil obtenção; e iii) ser

baseado em um método de predição simples, como é o caso da regressão linear.

Como trabalhos futuros, pretendemos implementar o trabalho de Piga et al. com o intuito de compará-lo com o nosso trabalho sob as mesmas condições. Pretendemos também avaliar o impacto do uso de VMs sob o consumo de energia do servidor. Por fim, também temos como meta o estudo de técnicas estatísticas mais elaboradas para a predição do consumo energético de outras cargas de trabalho também comum em *data centers* de computação em nuvem.

Referências

Barroso, L. A. and Hölzle, U. (2007). The case for energy-proportional computing. *Computer*, 40(12):33–37.

Bedard, D., Lim, M. Y., Fowler, R., and Porterfield, A. (2010). Powermon: Fine-grained and integrated power monitoring for commodity computer systems. In *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)*, pages 479–484.

Beloglazov, A. and Buyya, R. (2012). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurr. Comput. : Pract. Exper.*, 24(13):1397–1420.

Bohrer, P., Elnozahy, E. N., Keller, T., Kistler, M., Lefurgy, C., McDowell, C., and Rajamony, R. (2002). The case for power management in web servers. In *Power aware computing*, pages 261–289.

Cochran, R., Hankendi, C., Coskun, A. K., and Reda, S. (2011). Pack & Cap: Adaptive DVFS and Thread Packing Under Power Caps. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 175–185.

collectd (2016). Collectd – The system statistics collection daemon. `https://collectd.org`. Último acesso: oct-14-2015.

David, H., Gorbatov, E., Hanebutte, U. R., Khanna, R., and Le, C. (2010). Rapl: Memory power estimation and capping. In *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*, pages 189–194.

Fan, X., Weber, W.-D., and Barroso, L. A. (2007). Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, pages 13–23.

Ge, R., Feng, X., Song, S., Chang, H.-C., Li, D., and Cameron, K. W. (2010). Powerpack: Energy profiling and analysis of high-performance systems and applications. *IEEE Transactions on Parallel Distributed Systems*, 21(5):658–671.

Hsu, C.-h. and Feng, W.-c. (2005). A power-aware run-time system for high-performance computing. In *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, pages 1–.

Isci, C., Buyuktosunoglu, A., Cher, C.-Y., Bose, P., and Martonosi, M. (2006a). An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 347–358.

Isci, C., Contreras, G., and Martonosi, M. (2006b). Live, runtime phase monitoring and prediction on real systems with application to dynamic power management. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 359–370.

Isci, C. and Martonosi, M. (2003). Runtime power monitoring in high-end processors: Methodology and empirical data. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 93–.

Joseph, R. and Martonosi, M. (2001). Run-time power estimation in high performance microprocessors. In *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, pages 135–140.

Kaxiras, S. and Martonosi, M. (2008). *Computer Architecture Techniques for Power-Efficiency*. Morgan and Claypool Publishers, 1st edition.

Menasce, D. A., Dowdy, L. W., and Almeida, V. A. F. (2004). *Performance by Design: Computer Capacity Planning By Example*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Metri, G., Srinivasaraghavan, S., Shi, W., and Brockmeyer, M. (2012). Experimental analysis of application specific energy efficiency of data centers with heterogeneous servers. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 786–793.

Mi, N., Casale, G., Cherkasova, L., and Smirni, E. (2010). Sizing multi-tier systems with temporal dependence: benchmarks and analytic models. *Journal of Internet Services and Applications*, 1(2):117–134.

Piga, L., Bergamaschi, R. A., and Rigo, S. (2014). Empirical and analytical approaches for web server power modeling. *Cluster Computing*, 17(4):1279–1293.

Rotem, E., Naveh, A., Ananthakrishnan, A., Weissmann, E., and Rajwan, D. (2012). Power-management architecture of the intel microarchitecture code-named sandy bridge. *IEEE Micro*, 32(2):20–27.

Rountree, B., Lownenthal, D. K., de Supinski, B. R., Schulz, M., Freeh, V. W., and Bletsch, T. (2009). Adagio: Making DVS Practical for Complex HPC Applications. In *Proceedings of the 23rd International Conference on Supercomputing*, pages 460–469.

Scaramella, J., Marden, M., Daly, J., and Perry, R. (2014). The cost of retaining aging it infrastructure. Technical report, International Data Corporation (IDC), Framingham, MA.

TPC (2016). TPC-W Benchmark. `http://www.tpc.org`. Último acesso: oct-14-2015.

Wang, Q., Kanemasa, Y., Li, J., Lai, C. A., Matsubara, M., and Pu, C. (2013). Impact of DVFS on N-tier Application Performance. In *Proceedings of the First ACM SIGOPS Conference on Timely Results in Operating Systems*, pages 5:1–5:16.

Zhang, Q., Cherkasova, L., and Smirni, E. (2007). A regression-based analytic model for dynamic resource provisioning of multi-tier applications. In *Proceedings of the Fourth International Conference on Autonomic Computing*, pages 27–.

Using Load Shedding to Fight Tail-Latency on Runtime-Based Services

Daniel Fireman, Raquel Lopes, João Brunet

[1]Departamento de Sistemas e Computação
Universidade Federal de Campina Grande (UFCG)
Caixa Postal 10.106 – 58.109-970 – Campina Grande – PB – Brasil

danielfireman@lsd.ufcg.edu.br, {raquel, joao.arthur}@computacao.ufcg.edu.br

Abstract. *HTTP services written in managed runtime languages such as Java are popular nowadays. By relying on a runtime environment (RE), these services can benefit from safer code, cross-platform, etc. However, it is well known that RE's pauses due to garbage collection increase response times (i.e. tail latency). As modern services often rely on many remote calls, the overall performance turns out to be determined by the tail, instead of the average latency. To address this problem we propose an easy-to-use combination of load shedding and control of garbage collector interventions. We implemented and evaluated a prototype in Java. Our results show a reduction of tail latency by approximately* 40%*, while throughput and CPU utilization were negligibly impacted.*

Resumo. *Serviços HTTP escritos em linguagens baseadas em runtime são muito populares nos dias de hoje. Por se basearem em uma runtime, estes serviços podem usufruir de benefícios como código mais seguro, multiplataforma e etc. Todavia, é bem conhecido que pausas da runtime devido à coleta de lixo aumentam o tempo de resposta. Como serviços modernos frequentemente se baseiam em diversas chamadas remotas, o desempenho total do sistema acaba sendo determinado pela cauda da distribuição de latência, ao invés da sua média. Como solução para esse problema, propomos uma combinação de prevenção de carga e controle das intervenções causadas por coletores de lixo. Nós implementamos um protótipo em Java e avaliamos o seu desempenho diante de workloads sintéticas. Os resultados mostram uma redução de aproximadamente* 40% *na cauda de latência, mantendo praticamente a mesma vazão e utilização de CPU.*

1. Introduction

Imagine a client making a request on a single web service. Ninety-nine times out of a hundred that request will be returned within an acceptable period of time. But one time out of hundred it may not. If you look at the service's latency distribution, there is one large entry at the tail end. When the service does not send upstream requests, all it means is that one client gets a slower response occasionally.

Now, instead of one, let us imagine that a request will require response from 100 services. That changes everything about your system's responsiveness. Suddenly the majority of queries (i.e., $1 - 0.99^{100} = 63\%$) will take greater than 1 second. Hence,

temporary high-latency episodes may come to dominate overall service performance at large scale [Dean and Barroso 2013].

It is important to point out that, not only the temporary high-latency is a problem, but also the high variability on the tail of the distribution because it increases the unpredictability of the system latency. To better explain this problem, let us analyze Figure 1. It shows a concrete example of the service time's percentiles for a synthetic workload. For example, looking at the 99.9 percentile line, a point in the graph means that, at a given time (horizontal axis), 99.9% of the requests are below the corresponding latency (vertical axis). As expected, the more we increase the percentiles, the slower are the service times.

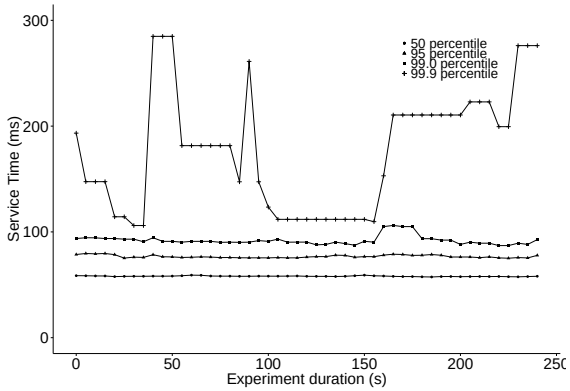


Figure 1. Service time of a single experiment run.

For clarity purposes, we would like to point out two main aspects of Figure 1. First, as you can see, the 99.9 percentile latency is substantially worse in comparison to the other percentiles. Also, the variability of the 99.9 percentile is substantially greater than the others. In this paper, we propose a solution to narrow down the 99.9 percentile line (reduce its variability) and bring it closer to the other percentiles (reduce service time tail) without significantly compromising the throughput and CPU utilization of the service.

To achieve this, we first had to understand the state-of-the practice on web services and what causes the aforementioned problem. In this context, there has been an increasing push for low latency and variability at the tail [Dean and Barroso 2013, Rumble et al. 2011]. To get a real feeling about these large scale services, a single Facebook page load can involve fetching hundreds of results from their distributed caching layer [Nishtala et al. 2013], while a Bing search consists of 15 stages and involves thousands of servers in some of them [Jalaparti et al. 2013]. These applications require latency in microseconds with tight tail guarantees.

Also, a large portion of workloads that are running in cloud datacenters are written in programming languages supported by managed runtime systems[Meyerovich and Rabkin 2013]. Companies such as Twitter [Humble 2011] and Facebook [Verlaguet and Menghrajani 2014] are writing most of their code in Scala and PHP. At the same time, cloud platforms such as Google AppEngine [Krishnan and Gonzalez 2015] and Microsoft Azure [Li 2009] are supporting managed languages as explicit targets. Finally, many web startups write their code in languages

such as Python, Ruby or JavaScript (which are all managed languages), as it allows them to iterate quickly. All these facts confirm that managed runtimes are popular and it does not seem a temporary situation.

Managed languages comes with a price though. Unfortunately, garbage collection (GC) is one of the main causes of high tail-latency [Dean and Barroso 2013]. That happens mainly due to two reasons: i) stop-of-the-world pauses [Gidra et al. 2013], in which applications are completely stopped during collection, and ii) CPU competition, as GC threads could run concurrently with the application.

In summary, managed languages are popular but the GC causes high tail-latency, as the ones observed in Figure 1. In order to eliminate garbage collection related latency with minimum impact on throughput, we propose an easy-to-use HTTP request interceptor to be added to web services in managed languages. Inspired by [Terei and Levy 2015], Garbage Collector Control Interceptor (GCI) controls garbage collection interventions and uses built-in HTTP load shedding mechanisms to deal with the requests that arrives during these interventions. We claim that the proposed solution has the following characteristics:

- Is application code and load agnostic;
- Requires little or no configuration;
- Requires no understanding of the runtime system's internals;
- Relies on vanilla HTTP for load shedding (RFC7231);
- Is completely open source.

We implemented a Java prototype of GCI to reduce garbage collection related latency in a stateless HTTP service. We compare the service performance with GCI on and off (baseline), both cases using the OpenJDK's default garbage collector scheme. Our results show that activating GCI leads to a 40% reduction in 99.9 percentile latency and a 5 times reduction in 99.9 percentile latency variability. All those benefits coming with a 1.2% decrease in throughput and 0.5% decrease CPU utilization in the worst case.

The remaining of this paper is organized as follows. Section 2 describes some key concepts. In Section 3 we describe the design and implementation of GCI. In Section 4 we describe our research questions and detail the performed evaluation and results. In Section 5 we describe related work and finally, we conclude in Section 6.

2. Background

2.1. Datacenter workload expectations

Nowadays, the workload running inside a datacenter is very heterogeneous, spanning from batch jobs that need no quality of service (QoS) guarantees to distributed HTTP services which are typically user-facing applications. These services are formed by many interacting web services that are lightweight, maintainable, and scalable [Rodriguez 2008]. Since these jobs are user-facing, they need QoS guarantees, especially regarding their latency. A single user request may travel among various service points before returning to the user again, touching dozens or even hundreds of servers. To allow these interactions between services without impacting the latency experienced by users, it is of utmost importance to consider not only the average latency of these services, but also their latency at the tail. In this environment, it is important to guarantee that even the 99.9 percentiles of the latency of the services are limited by rigid bounds [Terei and Levy 2015].

2.2. Opportunities of Managed Languages

Many of these web services are built using managed languages, i.e., languages whose programs execute on top of a runtime environment. Current examples of such languages are Java, Python and Ruby. The main reason of the popularity of these managed languages is probably time to market, since they often offer to developers some facilities such as dynamic typing, class resolution at runtime, reflection and garbage collection. Further advantages of managed languages include opportunities for dynamic optimization, especially when we consider long running services[Dean and Barroso 2013]. Jobs that run for a long time amortize the startup time delays impressed by the just-in-time compilation. Many managed runtime systems also compact memory during execution, which is important for long-running applications. This compaction avoids performance degradation from fragmentation and loss of locality.

2.3. State of Garbage Collection

Garbage collection is the automatic memory management mechanism of an executing program. It reduces the engineering overhead from explicitly dealing with pointers and eliminates many sources of errors. Even though runtime environment implementers have been working on building faster garbage collectors [Tene et al. 2011, Ugawa et al. 2014], pause times at the tail are still too long[Blackburn et al. 2008]. This is an important issue for those managed languages, since the impact of the garbage collection is often unpredictable and hard to debug. For instance, GC performance can vary greatly from system to system, or even over the lifetime of a single system [Soman et al. 2004].

As the GC impact is an unavoidable fact for those applications running on top of a runtime environment, it is important to have external tools to deal with it. Many languages provide support for programmable interfaces to interact with the garbage collector. For instance, Java Virtual Machine (JVM) supports the $System.gc()$ function, which suggests to the runtime to start the GC. It also has the Garbage Collection Notifications API extension, which supports notifying the application after a collection has completed[Oracle 2015]. JVM has no support to automatically disabling garbage collection. Microsoft .NET also has a Garbage Collection Notifications API, which offers a broad set of notification options [Microsoft 2015]. It also supports forcing a collection through $GC.Collect()$ function and disabling GC. We are aware of similar APIs in many other languages, for instance Python, Ruby and Go. Without these APIs the GCI would not be possible.

3. Garbage Collection Control Interceptor (GCI)

Changing the application code to deal with runtime pauses is difficult, as the GC behavior is often unpredictable and its impact is hard to debug. Furthermore, tuning the GC of a production system is hard. On the one side, it depends on characteristics of the load the service is subjected to, which can be very dynamic in worldwide distributed systems. On the other side, it depends on the service code, which can be deployed many times a day through continuous delivery pipelines.

To help on that matter we propose the Garbage Collection Control Interceptor (GCI). GCI is an HTTP request interceptor which controls garbage collector interventions and take action based on those events. For the purposes of this work, we define an

HTTP request interceptor as a piece of code that is executed before every single request received by an HTTP service. The concept itself is widely supported across most HTTP server frameworks, but its name can vary. Frameworks like Ruby on Rails, Java's J2EE and Spring call them Filters, whereas it is called Middleware amongst Go developers. Typically, interceptors can be activated with very minimum code changes (usually one line), offering a non-intrusive way of performing common processing desired for every HTTP request.

It is important to note that GCI is not an approach to garbage collection, but a new approach to dealing with its performance impact on HTTP services. GCI is a technique that is agnostic with regards to the HTTP service code and its load. It requires no (or very little) specific configuration or understanding of the runtime system's internals. As a consequence, it is easy to use. GCI relies on vanilla HTTP specification for load shedding (RFC7231) and is completely open source[1].

Part of the GCI request processing is to decide when GC must run. Our proposal is quite simple: monitor the utilization of the runtime's heap pool(s) and when it reaches a certain threshold, it is time to collect the garbage. With this in mind, Algorithm 1 shows the pseudocode of the interceptor.

Input: $Resp$: HTTP response which is going to be sent to the client
Output: Whether to continue the request handling process

```
1  begin
2      if not ShouldAcceptRequest() then
3          Resp.SetStatusCode(503)
4          Resp.SetHeader("Retry-After", EstimateUnavailability())
5          return False
6      end
7      if SampleHeapUsage() then
8          if GetHeapUsage() > SHEDDING_THRESHOLD then
9              StopAcceptingRequests()
10             concurrent
11                 WaitOutstadingRequests()
12                 GC()
13                 StartAcceptingRequests()
14             end
15         end
16     end
17     return True
18 end
```

Algorithm 1: Garbage Collector Control Interceptor Pseudocode

In summary, the GCI controls when GC must run, guaranteeing that no request competes with GC for CPU or is delayed by having to wait in queues during the GC intervention. The algorithm is simple and for practical purposes it can be implemented to work with Java, Go, Ruby, and Python runtimes (but not restricted to them).

The processing of every request starts by checking whether the incoming request

[1] Available at https://github.com/danielfireman/gci

should be processed. If the request is allowed to be processed, the next step is to verify whether the usage of memory pools reached the specified threshold. If so, the system stops receiving new requests and a concurrent code block starts (so the request being executed is not blocked). This concurrent block waits for all the outstanding requests to be processed to then and forces the activation of the garbage collector. When the collection finishes, the system starts accepting requests again. We are going to dive into algorithm details in the next subsections.

3.1. Shedding requests

When a request must not be accepted ($ShouldAcceptRequest()$ returns false), the response is modified and the interceptor method returns false. This returned value determines the end of the request processing and immediate delivery of the HTTP response. The modified response has a 503 status code (Service Unavailable). As per RFC7231, the service unavailable status indicates that the service is currently unable to handle the request due to a temporary overload or scheduled maintenance, which will likely be alleviated after some delay [Fielding and Reschke 2014].

Furthermore, GCI shed responses always have the Retry-After header set. The service unavailability duration can be estimated using linear extrapolation from previous events. This approach uses the values of the previous utilization of target memory pools as predictors (or the overall heap, for non-generational GC schemes).

Shed requests may be immediately resent to another server. In practice, this is done transparently by client APIs or load balancers. Also, the Retry-After response header field suggests an appropriate amount of time for the client to wait before sending requests again to the unavailable service.

3.2. Sampling Memory Usage Check

All languages considered export methods to fetch information about memory utilization. Even though checking memory usage does not incur in a prohibitive cost when done once in a while, check memory usage at each request would make GCI unusable, especially in high load production deployments as ours. To decrease this overhead, GCI uses a sampling window. Thus, the GCI knows the memory utilization information related to the most recent window.

The $SampleHeapUsage()$ method checks the sampling window and returns true whether it is time to perform a memory utilization check, i.e, if a new window must be considered. The size of this window varies based on the previous number of requests processed between consecutive garbage collections. Using a number of requests is better than a time interval for two reasons. Firstly, it is less impacted by load peaks and, secondly, it does not trigger unnecessary checks due to load valleys. All these values related to the memory utilization monitoring are configurable.

3.3. Controlling Garbage Collector Activity

When the shedding threshold has been reached, no more requests can be accepted until garbage collection operation finishes. This is done by calling the $StopAcceptingRequests$, which makes $ShouldAcceptRequest$ return false until $StartAcceptingRequests$ call.

Garbage collector activity might incur in CPU competition or stop-of-the-world pauses. For this reason, it is important to ensure that garbage collection does not occur while processing requests. Thus, GCI must wait for all outstanding requests to finish before collecting the memory garbage. Furthermore, these actions must not block the request being processed (which would incur in latency increase). This is the reason for the $concurrent$ block (lines 10-14), which could, for example, be implemented in Java by running the code in a new thread.

GCI's garbage collection control depends on: i) triggering (forcing) a garbage collection and ii) avoiding automatic garbage collection. The former is expressed in the pseudocode as $GC()$. As an example, the Java implementation of the $GC()$ method can be executed through a $System.gc()$ call. All target languages expose similar functions to trigger garbage collections.

Disabling automatic garbage collections can be a problem for languages like Java, which lack of programmatic ways of doing so. One way to deal with this limitation is to make sure that automatic GC interventions will be as infrequent as possible. How to do that? Since automatic GC occurs when the memory fills up, all we need to do is to configure the runtime with the maximum possible heap and/or memory pools (for generational GCs). Fortunately, other languages like Python, Ruby and Go have easy ways to entirely disable automatic garbage collection, thus not requiring this workaround.

It is important to notice that GCI relies on the garbage collection scheme available. It is orthogonal to any GC specifics as well as to any custom configuration or tuning performed.

4. Evaluation

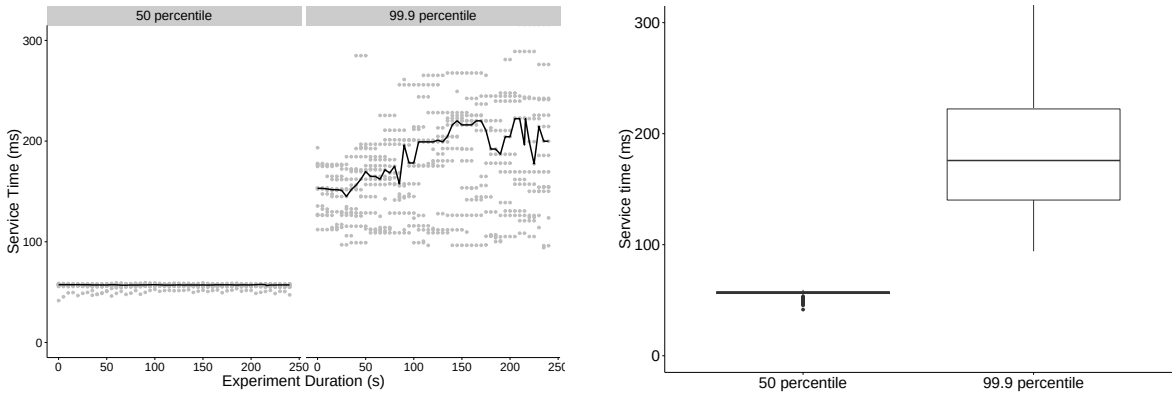
4.1. Experimental Setup

Let us start the evaluation by remembering the two main problems GCI aims to solve. First, Figure 2(a) illustrates service times for a synthetic workload over time. The black line is the average service times (median). The difference between the 99.9 and 50 percentile ranges from $72.69ms$ to $200.20ms$, which represents a meaningful quality of service degradation.

Second, in Figure 2(b), we present box plots that summarize the variability of both 50 and 99.9 percentile cases. The interquartile range (IQR) of the 99.9 percentile is $70.4ms$, which is in contrast with 50 percentile IQR (close to zero). That makes service time's predictability very difficult for the tail end, affecting capacity planning and SLOs definition, for example.

We conducted a 1-factor design experiment to evaluate Garbage Collector Interceptor impact on the 99.9 percentile service time and its variability. The activation of GCI was the only factor (GCI On and Off) and the service time was the dependent variable. We also chose a simple service request handling flow: a CPU intensive operation and small pause, simulating a blocking I/O call. This flow would represent, for instance, a query to a database and processing results before sending the response back to the client.

We replicated each experiment 15 times. Each one of them last around 8 minutes. We discarded the first 4 minutes of each experiment to minimize JVM warm-up



(a) 50 and 99.9 service time percentiles (b) Variability summary

Figure 2. Average service time

effects [Blackburn et al. 2008]. A client running on a different machine generated a constant load of 70 requests per second. Service times were measured for later evaluation.

The HTTP server executed in a virtual machine with 2 VCPUs (2660 MHZ) and 4GB of RAM running on a Ubuntu Linux (kernel version 4.4.0-53-generic). The server was executed using the OpenJDK 1.8.0_11 64-Bit Server VM (build 25.111-b14). Furthermore, to activate GCI in Java we needed to avoid as much as possible the automatic garbage collection. Because of that, we fixed heap size to 1 GB (i.e., setting -Xms1024m and -Xmx1024m) and split the heap equally between young and old generations pools (i.e. -XX:NewRatio=1).

4.2. Service Time Improvements

As we expect GCI to narrow and decrease the service time distribution tail, we drove our evaluation based on the following null hypotheses:

$H_{0,1}$: GCI does not improve the 99.9 percentile of the service time.

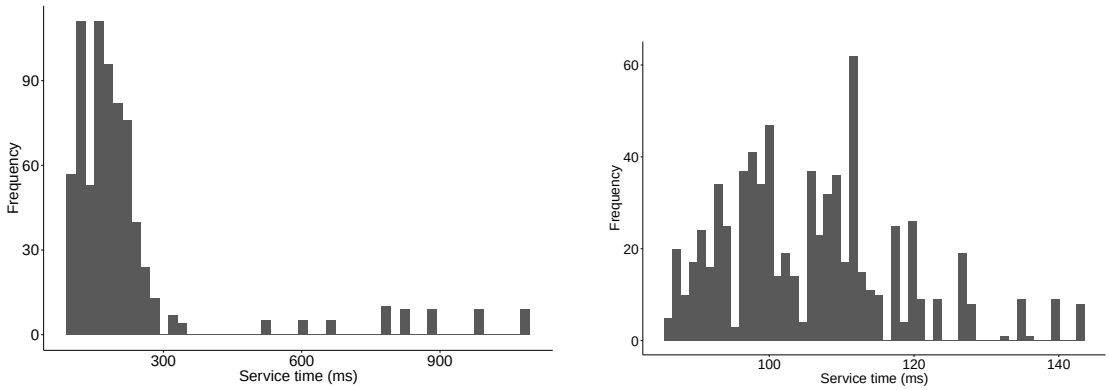
$H_{0,2}$: GCI does not reduce the 99.9 percentile variability of the service time.

We measured the 99.9 percentile service time with both GCI off and on (Figures 3(a) and 3(b)) to address these hypotheses.

Histograms in Figure 3 give the shape of the 99.9 percentile distributions of the service times. Please, be aware that axes are different. As you can note, the GCI reduces the tail of the 99.9 service time percentile, as the distribution becomes more symmetric. Furthermore, the GCI decreases the service time variation, once the range is narrower (from $]0, 1200]ms$ to $]0, 150]ms$).

In summary, we could say that GCI leads to the following benefits:

- Faster service times: within the 99.9 service time percentile, the median decreased from $175.8ms$ to $105.2ms$ and
- Predictable service times: within the 99.9 service time percentile, the IQR went down from $82.1ms$ to $14.7ms$. With less variation it easier to predict even the maximum expected service times.



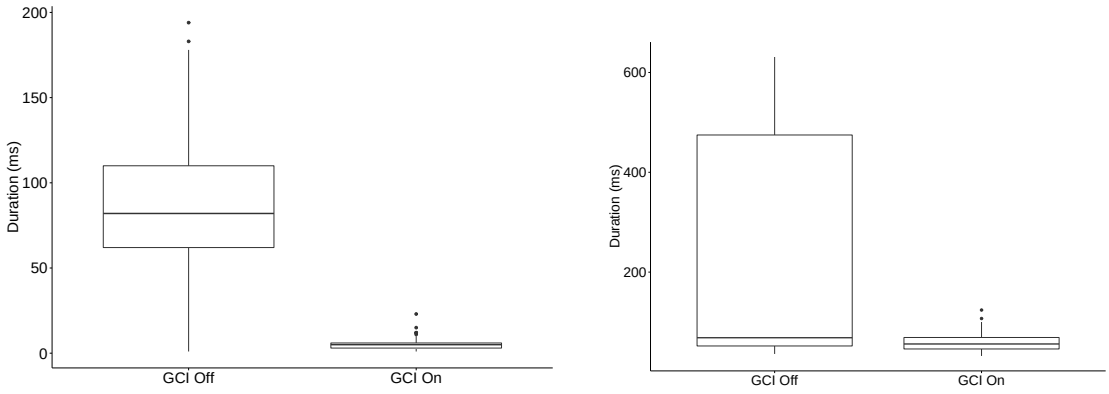
(a) GCI Off (b) GCI On

Figure 3. Histogram of 99.9 service time percentile

To statistically confirm these results, we carried out one-sided version of the non-parametric Mann-Whitney U Test [Hettmansperger 2011]. We chose this test because both samples do not come from a normal distribution, as confirmed by the very low p-values of Shapiro-Wilk Test (8.37^{-16} and 1.005^{-14}) [Shapiro and Wilk 1965]. Based on the result of the Mann-Whitney U test (p-value $< 2.2^{-16}$), we refute hypothesis $H_{0,1}$, with an estimated improvement around $71.42ms$. As for hypothesis $H_{0,2}$, we had already shown evidences of its refutal based on the I.Q.R. improvement.

4.3. Understanding the Impact on GC Behavior

As GCI controls garbage collections and sheds requests, it ends up changing GC's throughput and footprint [Sun Microsystems 2009]. To illustrate that, we present in Figure 4 an aggregated summary of GC interventions considering all experiment runs with GCI on and off.



(a) Minor GCs (b) Major GCs

Figure 4. Aggregated garbage collection activity

We analyzed the time span of the two garbage collection types: major and minor. The Java runtime's heap is managed in generations (young and tenure), which are memory pools holding objects of different ages. When a generation is full, the JVM triggers

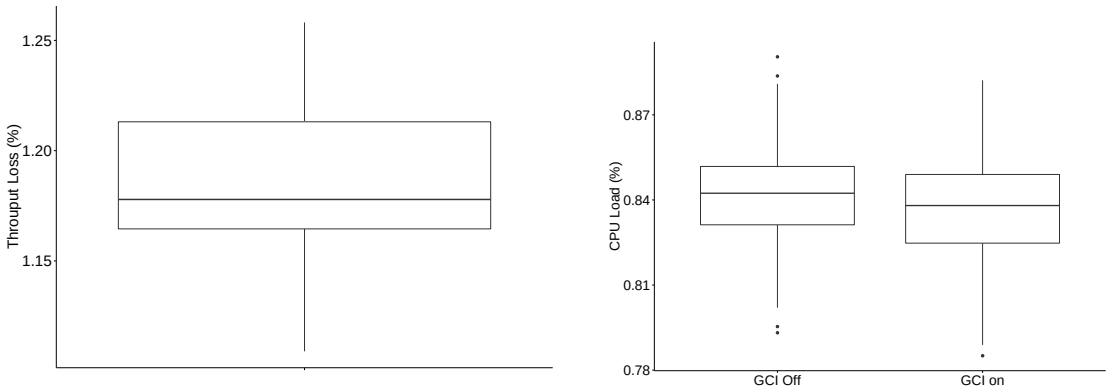
garbage collection. For the young generation, it causes a minor garbage collection. During this process, the JVM moves the surviving objects to the tenured generation. When, eventually, the tenured generation is full, the JVM triggers a major collection. Major garbage collections usually last much longer than the minor ones because a significantly larger number of objects are involved.

Minor garbage collections are represented in Figure 4(a) and it is easy to confirm that activating GCI leads to shorter garbage collections and also less variability.

Activating GCI also leads to shorter major collections and also less variability (Figure 4(b)). This happens because our solution triggers minor and major garbage collections when either pool reaches the shed threshold of utilization. By forcing garbage collection before the the automatic increasing of the generations size, GCI prevents long runs of the major garbage collection.

4.4. Overhead

We are aware that our approach deals with a trade-off: by shedding some requests to improve service time we decrease the overall service throughput. Naturally, our solution could not significantly compromise throughput or CPU load. For this reason, we also investigated the overhead that GCI generates. We calculated throughput loss as the ratio between shed requests and total requests (%).



(a) Throughput loss when GCI is on

(b) CPU Load

Figure 5. Throughput and CPU load aggregated summary

Figure 5(a) confirms that GCI does impact on throughput but not substantially. With 95% of statistical confidence, the mean throughput loss is $[1.16605\%, 1.209268\%]$. In a nutshell, GCI leads to less than 1.2% throughput loss on average. We consider this to be a good trade-off given the improvements in performance.

As GCI controls which requests to shed, one might wonder how it impacts on CPU utilization. Figure 5(b) presents data on this matter. As we can see, GCI leads to a small decrease in CPU utilization. To be more precise, with 95% of statistical confidence, the mean CPU load reduced from $[84.0\%, 84.26\%]$ to $[83.54\%, 83.79\%]$, which is a marginal impact.

4.5. Threats to Validity

This paper presents preliminary results of garbage collection interceptor. It is limited by some factors that can jeopardize external validity in the context of the experimental design. We are confident about our measurements: they were collected using proper instrumentation and we discarded warm up phase. However, we still must state the following threats to external validity:

- We are using one application. We do not know the extent to which the positive results of this study can be generalized to other applications;
- We focused on Java Virtual Machine. We are confident about the possibility of implementing our solution in other languages, but we cannot guarantee that the results of our study are extended to all such languages, especially when we consider the particularities of the garbage collection mechanisms implemented by other runtime environments;
- One machine configuration. For the sake of simplicity we considered only one type of server running the application. Different types of servers, with different memory sizes may lead to different results.

We hope to improve these threats in the future by considering other applications, languages and server configurations. It is important to mention that another threat would be that we fixed the workload to a constant load. In real environments, workloads are more dynamic and less predictable. However, we can consider that there is an auto scaling system adding and releasing resources from the application dynamically. That would guarantee that all the servers running are being highly and equally used, as the server we set up in our experiment.

5. Related Work

Some previous work also tackle the problem of high tail-latency due to the garbage collection done by the runtime environments. We are not aware, however, of a previous work that provides such simple and easy to use solution to deal with this problem by controlling garbage collection interventions and shedding the incoming requests during these interventions.

In [Terei and Levy 2015], Terei and Levy defined BLADE, which is an API that leverages existing failure recovery mechanisms in distributed systems to coordinate garbage collection and bound latency. They investigated two usage scenarios: an HTTP load-balancer and the Raft consensus algorithm. In both cases, latency at the tail using BLADE is up to three orders of magnitude better. In order to take advantage of BLADE, applications must be modified and use the BLADE API, which excludes all the legacy applications from bounding latency. When developing an application using BLADE API, developers need to know about memory management, garbage collection and other details. Furthermore, BLADE has only been implemented for Go language so far. Our solution pursues similar goals as BLADE, but by different means. By focusing on HTTP Services we can provide a much simpler and easy-to-use service, that plugs in the application, i.e. it is not part of the application by construction.

In [Maas et al. 2016] authors propose Taurus, which is a mechanism to reduce tail latency by coordinating garbage collection in a distributed system. Taurus is a JVM

replacement which can run unmodified Java applications and enforces user-defined coordination policies. They evaluated Taurus using two different applications: Spark [65] and Cassandra [33].

Both, BLADE and Taurus coordinate runtime activities considering the whole distributed application to avoid service disruptions. These systems leave developers with the task of programming or describing the coordination itself, which requires knowledge about the application, its workload, the environment, besides specific tuning. Another downside of Taurus is that it relies on a modified version of the JVM, which will need to be updated and maintained as any other component of the system. Our solution is orthogonal to these solutions, since it focus on coordinating each independent endpoint of the application independently by controlling GC executions and shedding requests during garbage collection interventions.

6. Conclusions and Future Work

This paper proposes Garbage Collection Control Interceptor (GCI) - a request interceptor which aims to control garbage collector interventions to improve its performance impact on HTTP services. GCI is a load and app-independent technique. It requires a marginal configuration effort and no understanding of the runtime system's internals. As a consequence, it is easy to use and port to other languages. GCI relies on vanilla HTTP specification for load shedding (RFC7231) and it is completely open source.

We evaluated GCI on a stateless HTTP Service with a constant workload. Experimental results showed that GCI significantly (40%) reduces 99.9 service time percentile and its variation (interquartile range decreased by 82%). That means that our work is a step towards faster and more predictable service times. Results also demonstrated a very small negative impact on throughput and CPU load.

Our main future work is to perform a broader evaluation of our approach. That includes measuring GCI impact on real world HTTP Services, such as Elasticsearch. It also includes to evaluate GCI implementations regarding other programming languages, such as Python, Ruby and Go. At last, even though GCI impact on throughput is marginal, we intend to act on the shed requests to reduce such impact to a minimal one.

Acknowledgments: This work was conducted during a scholarship supported by CAPES – Brazilian Federal Agency for Support and Evaluation of Graduate Education. This work is also sponsored by the agreement between UFCG and ePol/PF.

References

Blackburn, S. M., McKinley, K. S., Garner, R., Hoffmann, C., Khan, A. M., Bentzur, R., Diwan, A., Feinberg, D., Frampton, D., Guyer, S. Z., Hirzel, M., Hosking, A., Jump, M., Lee, H., Moss, J. E. B., Phansalkar, A., Stefanovik, D., VanDrunen, T., von Dincklage, D., and Wiedermann, B. (2008). Wake up and smell the coffee: Evaluation methodology for the 21st century. *Commun. ACM*, 51(8):83–89.

Dean, J. and Barroso, L. A. (2013). The tail at scale. *Commun. ACM*, 56(2):74–80.

Fielding, R. and Reschke, J. (2014). Rfc 7231 - http/1.1 semantics and content.

Gidra, L., Thomas, G., Sopena, J., and Shapiro, M. (2013). A study of the scalability of stop-the-world garbage collectors on multicores. In *Proceedings of the Eighteenth*

International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '13, pages 229–240, New York, NY, USA. ACM.

Hettmansperger, T. P. (2011). *Robust nonparametric statistical methods*. CRC Press.

Humble, C. (2011). Twitter Shifting More Code to JVM, Citing Performance and Encapsulation As Primary Drivers.

Jalaparti, V., Bodik, P., Kandula, S., Menache, I., Rybalkin, M., and Yan, C. (2013). Speeding up distributed request-response workflows. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 219–230, New York, NY, USA. ACM.

Krishnan, S. T. and Gonzalez, J. U. (2015). *Building Your Next Big Thing with Google Cloud Platform: A Guide for Developers and Enterprise Architects*. Apress, Berkely, CA, USA, 1st edition.

Li, H. (2009). *Introducing Windows Azure*. Apress, Berkely, CA, USA.

Maas, M., Asanović, K., Harris, T., and Kubiatowicz, J. (2016). Taurus: A holistic language runtime system for coordinating distributed managed-language applications. *SIGOPS Oper. Syst. Rev.*, 50(2):457–471.

Meyerovich, L. A. and Rabkin, A. S. (2013). Empirical analysis of programming language adoption. In *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications*, OOPSLA '13, pages 1–18, New York, NY, USA. ACM.

Microsoft (2015). .NET garbage collection notifications api.

Nishtala, R., Fugal, H., Grimm, S., Kwiatkowski, M., Lee, H., Li, H. C., McElroy, R., Paleczny, M., Peek, D., Saab, P., Stafford, D., Tung, T., and Venkataramani, V. (2013). Scaling memcache at facebook. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, nsdi'13, pages 385–398, Berkeley, CA, USA. USENIX Association.

Oracle (2015). Java JMX garbage collection notification api.

Rodriguez, A. (2008). Restful web services: The basics. *Online article in IBM DeveloperWorks Technical Library*, 36.

Rumble, S. M., Ongaro, D., Stutsman, R., Rosenblum, M., and Ousterhout, J. K. (2011). It's time for low latency. In *Proceedings of the 13th USENIX Conference on Hot Topics in Operating Systems*, HotOS'13, pages 11–11, Berkeley, CA, USA. USENIX Association.

Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 3(52).

Soman, S., Krintz, C., and Bacon, D. F. (2004). Dynamic selection of application-specific garbage collectors. In *Proceedings of the 4th International Symposium on Memory Management*, ISMM '04, pages 49–60, New York, NY, USA. ACM.

Sun Microsystems (2009). Java SE 6 HotSpot virtual machine garbage collection tuning.

Tene, G., Iyengar, B., and Wolf, M. (2011). C4: The continuously concurrent compacting collector. In *Proceedings of the International Symposium on Memory Management*, ISMM '11, pages 79–88, New York, NY, USA. ACM.

Terei, D. and Levy, A. A. (2015). Blade: A data center garbage collector. *CoRR*, abs/1504.02578.

Ugawa, T., Jones, R. E., and Ritson, C. G. (2014). Reference object processing in on-the-fly garbage collection. In *Proceedings of the 2014 International Symposium on Memory Management*, ISMM '14, pages 59–69, New York, NY, USA. ACM.

Verlaguet, J. and Menghrajani, A. (2014). Hack: a new programming language for HHVM.

Trilha Principal do SBRC 2017
Sessão Técnica 4
Redes Veiculares I

A QoE-aware Reinforcement Approach to Disseminate Warning Videos on LTE-VANETs

Carlos Quadros [1], **Aldri Santos** [2], **Denis Rosário** [1], **Eduardo Cerqueira** [1]

[1]Faculty of Computer Engineering and Telecommunication – UFPA
[2]Wireless and Advanced Networks (NR2) - Dept. of Informatics – UFPR

{quadros, denis, cerqueira}@ufpa.br, aldri@ufpr.br,

Abstract. *The wide range of Vehicular Ad-hoc NETwork's (VANETs) applications, e.g., real-time video dissemination, have made VANETs an interesting field of mobile wireless communication. Vehicle-to-Vehicle (V2V) communication enables users to distribute significant amount of real-time video traffic over VANETs and to alleviate congestion over LTE networks. However, the video delivery process considering an adequate Quality of Experience (QoE) in VANETs is a critical issue in both academic and industrial communities due to dynamic network topology, importance of video QoE, and broadcast nature of VANETs. This paper presents a Qoe-Aware Reinforcement approach to disseminate warning videos on LTE-VANETs, called QAR. It enables an enhancement for routing protocols in LTE-VANETs that takes advantage of a centralized architecture around the base station to improve the route management, and to provide video dissemination with QoE support. We analyze the performance of QAR by using NS-2 simulations and a realistic urban mobility model. Results show gains of QAR in comparison to existing proposals, where it achieves video dissemination with QoE support, less routing overhead, and robustness in LTE-VANETs.*

1. Introduction

Nowadays, after homes and offices, vehicles are the third place where people spend the most time daily. Along with this, the possibility of integration of information and communication technologies with transportation infrastructure and vehicles over an ad-hoc network environment called Vehicular Ad-hoc NETwork (VANET) has been broadly perceived by governments, manufacturers, and academia as a promising concept for future realization of Intelligent Transportation System (ITS) [Zaimi et al. 2016]. With videos currently accounting for more than half of the Internet traffic, new VANET applications ranging from multimedia safety and security traffic warnings to live entertainment and advertising videos have become a trend and are increasingly present [Quadros et al. 2016].

Traditional VANET consists of Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications supported by wireless access technologies, such as IEEE 802.11p. Unlike V2V, V2I considers not only vehicles but also roadside units. However, the roadside units' deployments are very expensive, thus, modern vehicles are also envisioned to be equipped with different wireless access technologies, including interfaces to cellular communication, e.g., Long Term Evolution (LTE). In this way, a heterogeneous integration of VANETs with cellular mobile systems or connecting vehicles directly to LTE networks (LTE-VANETs) have been gaining a great attention over the past

few years. In several studies, authors have considered direct communication between cars and the LTE network [Ucar et al. 2016, Salvo et al. 2016].

Existing cellular network infrastructure, e.g., base stations (eNBs in LTE), can be employed to achieve V2I communication [Salvo et al. 2016]. However, warning video dissemination, which relies on a centralized LTE-VANET heterogeneous network must reach as many as possible vehicles going to the crash area and, at the same time, cope with other LTE traffics, e.g., Human-to-Human (H2H) and Floating Car Data (FCD) traffic. In this way, V2V communication represents a viable way to distribute significant amounts of live videos over LTE-VANETs and also alleviate congestion periods over LTE networks. In such scenario, the V2I communication takes place only for signaling and coordination of routes. As regards to the V2V communication, many challenges arise. Due to ad-hoc environment and the highly dynamic topology, connection interruptions can become frequent, very close vehicles can transmit the same packets to each other or even increasing congestion periods, etc. This all leads to an unnecessary data traffic growth, causing different impacts on the video quality and hindering action of drivers and rescue teams watching the received warning videos. Thus, dissemination of real-time video content with Quality of Experience (QoE) support is not a straightforward task.

Maximizing the user's QoE and dealing with the high vehicle mobility becomes an intricate task and has not been taken into account in most of studies to date [Quadros et al. 2016]. While QoS concentrates only on transmission statistics and network-based management, the QoE levels of the disseminated videos are associated with the subjective admeasurement of users, being key to the acceptable reception of videos [Zaimi et al. 2016]. With the presence of a centralized entity, such as LTE eNBs, it is possible to manage the broadcast routes of the videos, allowing routes to be altered as soon as data transmission problems are perceived, thus, keeping the video streaming without interruptions [Ucar et al. 2016]. In this way, the LTE-VANET routing service must be aware of QoE requirements and network conditions to recover or maintain video quality with a low overhead and high reachability in the disseminated area.

This paper presents a Qoe-Aware Reinforcement (QAR) approach to disseminate warning videos on LTE-VANETs. It enables an enhancement for routing protocols in LTE-VANETs by taking advantage of a centralized architecture around the eNB to improve route management and to provide video dissemination with QoE support. QAR uses eNBs and a traffic management server for the collection of localization and video-related parameters. Based on these data, the routing service selects the best forwarding nodes for the video dissemination. Aiming to the maintenance of routes that offer better QoE for video dissemination, QAR combines video-related parameters (e.g., different frame importance, frame position, and video distortion estimation) as well as positioning information to establish trade-offs between QoE and required hops. QAR leverages the LTE network present, admitting easily integration with different routing protocols, maintaining the packet delivery ratio, reacting positively to dynamic environments and enhancing the QoE level of the disseminated warning videos when compared to non-QoE-aware schemes. We add QAR to a straightforward distance-based protocol and evaluate its performance. Results show gains of QAR in comparison to existing proposals, where it achieves video dissemination with QoE support, less routing overhead, and robustness in LTE-VANETs.

We organize this paper as follows. Section 2 outlines the related work. Section 3 introduces the QAR approach. Section 4 shows simulation setup and results comparing current LTE-VANET-based works and QAR. Lastly, Section 5 presents the conclusions.

2. Related Work

Recently, many authors have proposed routing protocols for LTE-VANETs. LTE provides data dissemination to many users over a geographical area at fine granularity. Most of current schemes assume that vehicles transmit application data directly to eNBs or through clustering schemes (i.e., cluster members communicate with cluster heads by using IEEE 802.11p and cluster heads communicate with eNBs by using LTE). [Ucar et al. 2016] presented a hybrid architecture, namely VMaSC-LTE, which combines IEEE 802.11p clustering and LTE with the goal of achieving high packet delivery rate with a minimum usage of the LTE infrastructure. [Salvo et al. 2016] proposed a FCD off-loading scheme via clustering formation in a LTE-VANET. [Jia et al. 2014] introduced a Markovian-based model to mitigate FCD transmissions. These works aim to reduce the traffic rates transmitted over the LTE, however, they do not examine transmissions of long data traffic, e.g., real-time warning videos, and do not consider video-related parameters for decision-making. Besides, a pure LTE based architecture is not feasible for vehicular communication due to the overload of the eNBs by packets coming from a high vehicle traffic density, which directly impacts other LTE flows, e.g., FCD and H2H [Salvo et al. 2016].

In this way, hybrid architectures represent a viable way to transmit warning videos over LTE-VANETs, and to alleviate congestion over LTE networks. The V2I communication proceeds only for signalling and route management and discovery, while V2V communication proceeds for the video distribution. There are three strategies for data dissemination used to design many routing protocols, including hybrid schemes with topological approaches: distance-, location-, and counter-based [Gonzalez and Ramos 2016]. In these strategies, nodes decide by themselves if they further broadcast data or not through a distributed backoff phase (i.e., by comparing a locally measured value). In distance-based protocols, forwarding node candidates use only the distance to the farthest 1-hop neighbor from whom the packets has been sent as a proxy for rebroadcasting [Chang and Lee 2015, Slavik et al. 2015]. In location-based protocols, nodes share location information to allow retransmissions in the uncovered areas [Mir et al. 2016, Husain and Sharma 2016]. Lastly, in the counter-based protocols, nodes count the number of times that each packet is received during the backoff time to know the number of neighbors that so far have retransmitted the packets [Chekhar et al. 2016, Torres et al. 2015].

[Slavik et al. 2015] presented a Distance-to-Mean (DTM) method, which extends the distance-based strategy, where nodes rebroadcast packets as soon as they cover a large amount of physical area that neighboring nodes have not covered. Likewise, [Torres et al. 2015] introduced an Automatic Copies Distance-Based (ACDB) mechanism, which extends the Counter-based strategy to cope with variable vehicle density situations. Despite ACDB applies Peak Signal-to-Noise Ratio (PSNR) to assess the QoE of received videos, the main drawback of the above approaches consists of their reliance on a single positioning parameter to compute the backoff phase. This issue reduces the network reliability for long data transmission, e.g., live videos. Further, PSNR evaluation does not correlate well with the subjective acceptability of the users [Quadros et al. 2016]. Using a decentralized organization is totally justified in these works, since authors assume

that vehicles have a single IEEE 802.11p interface, and no cellular network interface. However, the latest technological advances enable vehicles to be equipped with multiple types of wireless interfaces, forming a multihomed heterogeneous network environment.

From our analysis, there are several approaches for video dissemination in LTE-VANETs. For V2V communication, approaches where nodes decide by themselves if they must retransmit data or not, are promising, since vehicles do not flood messages proactively, avoiding routing overhead. With vehicles equipped with LTE interfaces, it is possible to improve the management of flow dissemination. Further, existing proposals do not apply video-related parameters to reinforce the video dissemination, neither offer this key feature in an unified routing protocol so far, lacking of robustness and QoE-awareness.

3. The QoE-Aware Reinforcement (QAR) Approach

This section presents the QAR approach to reinforce the dissemination of warning videos in LTE-VANETs. It relies on a centralized structure around the VANET area, avoiding neighboring information exchange. QAR works jointly with an underlying routing protocol and intends to select forwarding nodes with high reachability, i.e., nodes that deliver videos to as many destinations in a physical area as possible. This refrains unnecessary routing overhead and impact on the existing LTE traffic, i.e., FCD and H2H. Further, QAR allows video dissemination with QoE support even in face of dynamic topology scenario changes. QAR runs on the infrastructure to establish QoE-aware routes for video dissemination, where it considers vehicles' location and video-related parameters for forwarding decision. Depending on the routing protocol, QAR can combine other different parameters, aiming at a more in-depth decision process, e.g., link quality, speed, etc.

QAR works in two phases, namely Contention-Based Forwarding (CBF), and Centralized QoE-supported Management (CQM) phases. In the CBF phase, the Source Node (SN) indicates through its LTE network interface that it will begin a video broadcast and it starts the video flooding. Thereon, the Forwarding Node Candidates (FNCs) compete with each other to participate in the transmission routes (i.e., Forwarding Nodes - FNs). The FNCs take into account positioning and video-related parameters for forwarding decision, where the winning nodes retransmit the video sequences further. The CQM phase considers V2V forwarding and V2I management, exploiting the previously built multi-hop paths in CBF, and enabling dynamic changes to other FNs in case of link failures or low QoE detection. We will detail each phase in the following subsections.

In our network model, let us suppose a scenario of warning video dissemination in cases of accidents or disasters, where vehicles or first responder teams, coming toward the crashed area, receive in advance the real-time videos of the accident from a SN. We consider k vehicles with identifiers ($i \in [1, k]$), moving over an multi-lane highway area. The combination of those nodes configures a graph $G(V, E)$, where vertices $V = \{v_1, v_2, ..., v_k\}$ mean a finite subset of k nodes, and edges $E = \{e_1, e_2, ..., e_m\}$ mean a finite set of asymmetric wireless links between them. We denote $N(v_i) \subset V$ as a subset of all 1-hop neighbors within the radio range of a given node v_i. Further, each node v_i has an IEEE 802.11p-compliant radio transceiver, through which it can communicate with $N(v_i)$, a LTE interface that allows each node v_i to be managed directly by a traffic server (nodes can also be managed by eNBs when it is possible to implement that directly in the eNB). Moreover, each v_i holds a GPS (to location awareness and synchronism in time),

a multimedia encoder/decoder, and a transmission buffer. Fig. 1 shows a simplified node architecture with QAR (Subsection 3.1), IEEE 802.11p and 3GPP LTE radios, and other interdependent elements of a node. Table 1 outlines the main symbols used in this paper.

Table 1. The main symbols used in the QAR description.

Symbol	Description
v_i	A node with an Id ($i \in [1, k]$)
$N(v_i)$	Set of 1-hop neighbors of v_i
SN	Source Node
FNC	Forwarding Node Candidate
FN	Forwarding Node
MS	Management Server
VF	Video Flow
$W(VF)$	Time Window to broadcast pkts
WZ	Warning Zone
FF	Fit Function
Pos_{param}	Positioning parameter of FF
Vid_{param}	Video-related parameter of FF

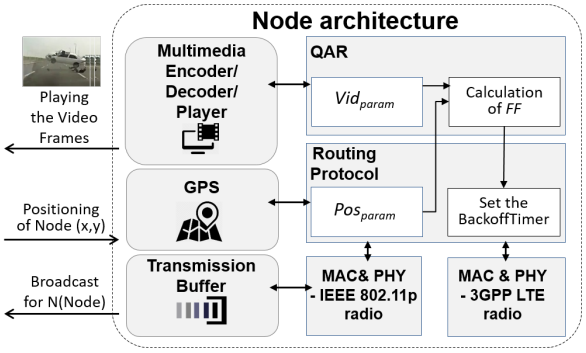


Figure 1. Simplified node architecture with QAR.

3.1. Contention-Based Forwarding (CBF)

At the beginning of the CBF phase of QAR, SN warns eNB that it will start a video dissemination through its LTE network interface. Particularly, SN sends a Notification to Video Broadcast (NVB) to a Management Server (MS), which assumes that SN is the initial node. The NVB contains the current SN position and a video Id $< SN_{pos}, SN_{Id} >$. Vehicles periodically send FCD to the MS in LTE, which contains information about presence, position, kinematics, or basic status [Jia et al. 2014]. The MS selects the FNCs that will compete to retransmit the video packets, based on SN positioning parameters and neighbors previously collected by FCD. Alg. 1 presents the CBF process performed at the MS. Firstly, MS obtains, from its FCD table, the current and past position of the broadcasting node (v_a) and its neighbor nodes $N(v_a)$ (Lines 1, 2, 4, and 5 of Alg. 1). In addition, MS establishes a Warning Zone (WZ) to limit the dissemination area, i.e., for a video exceeding a predetermined WZ, no further rebroadcast occurs (Line 3 of Alg. 1). We defined $WZ = 2km$, based on radio range of 250m in each hop and considering 8 as the maximum number hops for each path. This WZ provides a medium-long distance, as required in many rescue/disaster VANET scenarios [De Felice et al. 2015].

To limit the video dissemination for only vehicles going to the crash area, MS compares the angle between current and past position of v_a and current position of each $v_k \in N(v_a)$ to a threshold angle $\phi = 90°$ (Line 7 and 8 of Alg. 1). If the angle is greater than ϕ, MS performs a new comparison. The conditions mentioned above allow the system to mitigate the number of FNCs and the overhead of video packets by selecting only vehicles interested in the video. Thus, vehicles that have already passed by the accident area and vehicles going in the opposite direction will not need to receive the videos. The angle ϕ can be modified depending on the scenario, roads, or reachability rates of the routing service. As soon as the MS calculates the chosen FNCs, it sends an Authorization to Video broadcast (AVB) to FNCs and SN. In this way, the SN starts the broadcast of a Video Flow VF to its neighbors vehicles in a multi-hop fashion, i.e., SN broadcasts video packets p in a *Time Window*, denoted by $W(VF_i) \subset VF_i$, to all its neighbors ($N(SN)$). All nodes that were not chosen by MS to become FNCs, drop

$W(VF_i)$ at the MAC layer and, therefore, do not increasing network overhead. Through a contention distributed stage FNCs compete among themselves to choose which nodes will become Forwarding Nodes (FNs). Alg. 2 presents the process of CBF at the FNCs.

Algorithm 1 CBF phase in the Management Server (MS)

When a given node v_a (with $\mid N(v_a) \mid$ 1-hop neighbors) sends a NVB_i
$k \leftarrow 0$
Retrieves $v_a(x_{t-1}, y_{t-1})$, $v_a(x_t, y_t)$
if $\overline{v_a SN} < WZ$ **then** //Detecting if WZ is established
  while $k <\mid N(v_a) \mid$ **do** // $\forall v_k \in N(v_a)$
    Retrieves $v_k(x_{t-1}, y_{t-1})$, $v_k(x_t, y_t)$
    if $\angle v_a(x_t, y_t)v_a(x_{t-1}, y_{t-1})v_k(x_t, y_t) > \phi$) **then**
      if $\angle v_a(x_t, y_t)v_a(x_{t-1}, y_{t-1})v_k(x_{t-1}, y_{t-1}) \geq \angle v_a(x_t, y_t)v_a(x_{t-1}, y_{t-1})v_k(x_t, y_t)$ **then**
        $v_k \leftarrow$ FNC
        Sends AVB_i to v_k
      end if
    end if
    $k++$
  end while
  Sends AVB_i to v_a
end if

Algorithm 2 CBF phase in Forwarding Node Candidates (FNCs)

When a given node $v_b \in N(v_a)$ receives broadcasted packets ($W(VF_i) = \sum_{k=1}^{n} p_k$) from a node v_a:
if $v_b \supset AVB_i$ **then** // Detecting if v_b is a FNC.
  if $\exists!\ p_k \in W(VF_i)$ **and** $v_b \supset p_k$ **then** // Detecting redundant packets.
    Drop $W(VF_i)$ from v_b
    return
  else
    Sends NVB_i to MS
    Compute $FF(v_b)$ (Eq. (6))
    Start $BackoffTimer(v_b)$ (Eq. (1))
    while $BackoffTimer(v_b) \neq 0$ **do**
      if Overhear ! $p_k \in W(VF_i)$ **then** // Detecting redundant packets.
        if $\angle FNv_av_b < \lambda$ **then** // Angle comparison between FN and v_b.
          Cancel $BackoffTimer(v_b)$
          Drop $W(VF_i)$ from v_b **and** Cancel any new rebroadcast of $p_k \in W(VF_i)$
          return
        end if
      end if
    end while
    Rebroadcasts $W(VF_i)$ **and** $v_b \leftarrow FN$
  end if
end if

If v_b has already received an AVB_i from MS to forward the video packets and if $W(VF_i)$ contains only new received packets (Line 2 of Alg. 2), v_b sends a NVB_i to MS and apply the *Fit Function* (FF) (Eq. (6)) (Line 7 of Alg. 2). FF allows the network to mitigate the number of retransmissions and duplicated packets by choosing only the best FNCs. The value of FF $[0, FF_{max}]$ depends on parameters of the underlying routing protocol, such as positioning (Subsection 3.1.2), and video-related parameters, as shown in Subsubsection 3.1.1. Thus, after calculating the FF, v_b sets a $BackoffTimer$ according

to Eq. (1), and after the timeout, rebroadcasts the buffered packets ($W(VF_i)$). It is easy to see that nodes with higher values of FF are mapped to smaller $BackoffTimer$ values, and thus have a higher probability to forward the video packets.

$$BackoffTimer = CW_{Max} - FF \cdot (CW_{Max} - CW_{Min}) \quad (1)$$

Where CW $[CW_{Min}, CW_{Max}]$ is the size of the *Contention Window* in the 802.11p standard. The FNC that generates the smallest $BackoffTimer$ rebroadcasts $W(VF_i)$ first and becomes a FN (Line 18 of Alg. 2). Moreover, as expected in the IEEE 802.11p standard, vehicles are able to sense the channel during the $BackoffTimer$. Thus, in case of overhearing transmissions from another FN, $v_b \in N(FN)$ compares the angle between its own location, FN, and the sender node (v_a) to a threshold angle $\lambda = 45\,^{\circ}$. This threshold angle implies directly the reachability of the routing protocol (Line 11 of Alg. 2). When the angle between the previous selected FN and v_b is bigger than λ, v_b proceeds to retransmit $W(VF_i)$ itself, otherwise it remains silent. With this strategy, the received packets can be disseminated to other directions via multiple FNs.

In the next subsections, we will describe how QAR uses both, video-related and positioning parameters to compute FF and, thus, to choose the best FNs. The selected FNs retransmit the video flows to neighbors until WZ be reached and participate in the CQM phase, where dynamic routes of dissemination are built (Subsection 3.2).

3.1.1. Video-related parameters

Each packet p in a VF contains, in addition to the data payload, other encoder parameters, such as a frame-type flag, Id, length, timestamp, and packet segmentation. To obtain this information, the Deep Packet Inspection (DPI) algorithm enables extraction of the frame type and intra-frame dependency information for each p [Sherry et al. 2015], since each VF starts with a Group of Pictures (GoP) header and by one or more coded frames. The DPI methods have been used in existing works to collect information about malicious packets, frame type and intra-frame dependency, which are described in the video sequence and GoP headers [Quadros et al. 2016]. Thus, DPI is a good alternative to allow cross-layer multimedia networking solutions to improve the usage of network resources and the user's perception.

Regarding to the video structure, MPEG-4 video sequences are compressed in GoPs composed of I (Intra), P (Predictive), and B (Bidirectional) frames. Frames between two I-frames belong to one GoP, so that there is no fixed value for the GoP size (generally, 10-20 frames). I-frames are self-contained, however, to encode and decode P- and B-frames, the previous I-frame and/or P-frames in the same GoP are needed. Thus, when an I or P-frame is lost, all frames thereafter in the GoP become un-decodable, i.e., the same degree of packet loss may cause severe video quality degradation or may pass unnoticed, depending on which frame types are affected. Fig. 2 shows the different importance degrees for the user's perception in each frame type for an 18-frames GoP size. The length of a loss burst determines the number of subsequent frames in which this effect propagates. I-frames are the most important ones, followed by P-frames and, finally, B-frames (for a single B-frame lost, no impact is noticed visually, since no other frames are affected). Further, the loss of P-frames at the beginning of a GoP causes a higher video

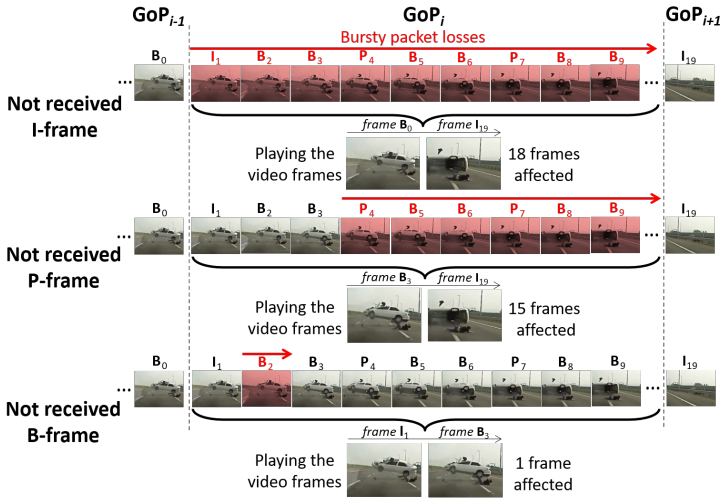


Figure 2. Different priority of frames.

distortion as detailed in [da Silva et al. 2016]. By considering the importance of each video frame, as well as the P-frame position within a GoP, QAR prioritizes frames with a greater impact on the average video distortion (σ_s^2). Thus, it assigns different weights to packets belonging to each frame as modeled by Eq. (2):

$$\sigma_s^2 \propto \begin{cases} \frac{\alpha_1(R_M - R_I)}{R_M} & \text{if s} \in \text{I-frame} \\ \frac{\alpha_2}{(2^{T-1}-1)R_M} \sum_{i=1}^{T-1} 2^{T-1-i}(R_M - R_{P_i}) & \text{if s} \in \text{P-frame} \\ \frac{\alpha_3(R_M - R_B)}{R_M} & \text{if s} \in \text{B-frame} \end{cases} \quad (2)$$

Where $T-1$ is the total of P-frames per GoP, R_I, R_{P_i}, and R_B mean the I, P (with position i in the GoP), and B-frame received rate in $W(VF_i)$, respectively. R_M is the maximum data-rate supported by the radio transceiver of each vehicle, e.g., for a DCMA-86P2 IEEE 802.11p Wi-Fi card, R_M = 6Mbps if P_{rx} > -93dbm [Quadros et al. 2016]. The parameters α_1, α_2, and α_3 are weighting factors, where $\sum_{i=1}^{3} \alpha_i = 1$.

The distortion model proposed by Shu Tao [Tao et al. 2008] considers the impact caused by the loss of packets of a video frame. Thus, for a video frame structure and a probability of occurrence of loss, Eq. (3) defines an overall distortion value for the whole stream, where L is the number of packets per frame obtained from the MPEG-4 configuration and packetization. The parameter $\overline{n}$ represents the loss burstiness ($1.06 \geq \overline{n} \geq 1$ for Bernoulli losses, depending on the aggressiveness of the burst errors). The attenuation factor γ ($\gamma < 1$) accounts for the effect of spatial filtering, and varies as a function of the video characteristics and decoder processing. P_e is the probability of loss events (of any length) in the video stream. Both, γ and P_e are given by the effect of the loss pattern experienced by the video stream and the codec's error concealment technique. Finally, the Mean Square Error (MSE) distortion $\overline{D}$ provides a QoE-estimate (Vid_{param}) by using a non-linear relation that measures the video quality level by comparing distortions caused by packet losses [Tao et al. 2008], according to each frame type, denoted in Eq. (4):

$$\overline{D} = L \cdot \overline{n} P_e \cdot \sigma_s^2 \cdot \left(\frac{\gamma^{-t+1} - (T+1)\gamma + T}{T(1-\gamma)^2} \right) \quad (3)$$

$$Vid_{param}^{v_b}(W(VF_i)) = \frac{1}{1 + exp(b_1 \cdot 10 \cdot log_{10}(255^2/\overline{D}) - b_2)} \quad (4)$$

Where, b_1 is the slope of the QoE mapping curve and b_2 is the central point. By considering 40 dB as the highest video quality, and the lowest video quality for values below 20 dB, the values of b_1 and b_2 are given by 0.5 and 30, respectively. Based on the average distortion caused by losses in the different frame types in $W(VF_i)$ it is possible for FNCs to compute a higher FF based on receiving the most important packets.

3.1.2. Coupling of the CBF phase of QAR with the Distance-Based Routing Protocol

This subsection presents QAR operating together with an underlying routing protocol. To assess QAR functionalities, we develop and adapt its CBF stage jointly with a straightforward broadcast protocol built using the distance-Based strategy, called DQAR protocol. Distance-based protocols calculate coverage through the distance (Pos_{param}) from the FNC to the previous sender node. When Pos_{param} is small, it means the FNC is close to the last sender, indicating it should not favor rebroadcasting. Only local positioning information is used in the distance-based strategy to calculate Pos_{param} in FNCs. Thus, the spatial distance is defined according to Eq. (5) for a FNC positioned at (x, y) and a previous sender node located at $(\overline{x}, \overline{y})$ and normalized to a value between zero and one by dividing by the maximum transmitting range (R) of the vechiles. Hence, FNCs with large geographical distance from previous sender node generate higher FF values.

$$Pos_{param} = \frac{1}{R}\sqrt{(x-\overline{x})^2 + (y-\overline{y})^2} \tag{5}$$

To add QAR with the distance-based protocol, we establish two parameters as input to FF (Eq. (6)): vehicle positioning (Pos_{param}) and video-related parameters (Vid_{param}), allowing a cross-layer selection of FNCs (Fig. 1) in addition to the only positioning parameters of the pure distance-based protocol. As defined in Eq. (5), the distance-based protocol does not exchange messages containing location or mobility information. Thus, DQAR also mitigates the overhead when determining the best FNC options (FNs). Further, the QAR steps can be easily adapted to other routing protocols by simply changing the parameters in the CBF stage for the FN selection process. It might be suitable for link quality-based, stochastic-based, or counter-based routing protocols.

3.2. Centralized QoE-supported Management (CQM)

As video transmissions have often long duration (e.g., 20 s), whenever a FNC wins the CBF phase, SN transmits video packets explicitly without any additional delay and in a pipeline fashion along the built route. Thereby, QAR reduces additional delays and packet duplication from the CBF phase by introducing the Centralized QoE-supported Management (CQM) phase. During the transmission, nodes must deliver the video content even in the presence of link failures or channel variations. QAR detects routing failures, providing a smoother route management by considering that every FN that composes the video dissemination routes should perceive whether it is still a reliable or valid route to transmit packets. This is achieved by receiving reply messages. We define a control packet, called Quality Warning Message (QWM), which contains the $\overline{D}$ perceived by each FN. Thus, if a FN receives a video flow, it must compute $\overline{D}$ (distortion) perceived in each $W(VF)$ and send a QWM to MS. A route in CQM returns to the CBF phase, when MS is notified that the video quality fallen below a predefined video distortion threshold. Further, MS

considers that the route is not valid anymore, as long as it does not receive any QWM from FNs within a certain period of time, i.e., timeout $= 0.5s$. Hence, it sends an AVB to the previous FN and the route returns to the CBF phase.

Upon computing Vid_{param} and Pos_{param}, each FN contains the two calculated parameters of the FF, i.e., $P = \{Vid_{param}, Pos_{param}\}$ ($\mid P \mid = 2$). Thus, considering the different weights $\omega_j \mid \sum_{j=1}^{\mid param \mid} \omega_j = 1$, Eq. (6) calculates FF by multiplying the values p_j in P and the weights of the evaluation parameters, similarly to others multi-criteria approaches [Jauregui and Malaina 2016]. From Eq. (6), other parameters can be added to FF of QAR, depending only on the underline routing protocol.

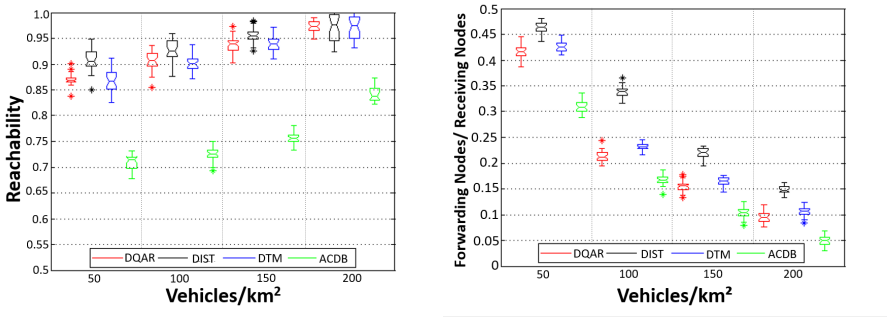
$$FF = \sum_{j=1}^{|P|} (p_j \times \omega_j) \tag{6}$$

4. Performance Evaluation

This section shows methodologies and metrics used to evaluate the transmitted video flows, and we compare the performance of DQAR with the main related works. To a proper scenario, we have considered a 10 Km portion of the San Diego Freeway, imported into Simulation of Urban MObility (SUMO), to reproduce the vehicle movements and interactions according to empirical data. In our simulations, vehicles move ranging from 20 to 30 m/s, each one holding an IEEE 802.11p (5.89 GHz, 6 Mbps) radio with about 250 m transmission range and a 3GPP LTE (700 MHz, 300 Mbps) radio with a transmission range up to 30km. We applied the Nakagami Fading Channel as propagation model. Similarly to [Torres et al. 2015], we scheduled an accident situation, so that when SN perceives this accident, it sends a NVB to MS and starts the broadcast of VF. Each VF must be received by vehicles within a $WZ = 2$km from SN, providing limitation of hops. The Radio Access Network (RAN) consists of eNBs, which manage radio resources and handover events. ENBs connect a Serving Gateway/PDN Gateway (SGW/PGW) via Evolved Packet Core (EPC) network, that contains a Mobility Management Entity (MME) with the MS, responsible for store vehicles' position information.

For the purpose of realistic results, we have adopted the EvalVid framework- A Video Quality Evaluation Tool-set that allows us evaluating the video quality. Thus, we have conducted the experiments by transmitting real MPEG-4 sequences (720 x 480 pixels) lasting approximately 20 s, available in [Video Sequences 2016a], with 768 kbps and 24 fps, internal GoP structure (size 20) configured as two B-frames for each P-frame. Finally, we added the videos and the road/vehicle features into Network Simulator 2.33.

To demonstrate the impact of DQAR (QAR coupled with a distance-based protocol), we used DTM [Slavik et al. 2015], ACDB [Torres et al. 2015], and a straightforward routing protocol built from the distance-based strategy (named DIST) for comparison. The DTM and ACDB protocols use distance-based and counter-based strategies, respectively. In DTM, the farther away the node to the spatial mean, the shorter the $BackoffTimer$. ACDB uses vehicles' density information to dynamically adjust a counter and the maximum FF before rebroadcasting packets. We adjusted these protocols with the CQM phase to reduce the contention phase: once a vehicle successfully transmits video packets, its timer for the next packets will be minimum. Further, we added each protocol to nodes with IEEE802.11p and LTE interfaces, where the eNB performs the



(a) (b)

Figure 3. Reachability and Forwarding Nodes by Receiving Nodes vs Veh/km²

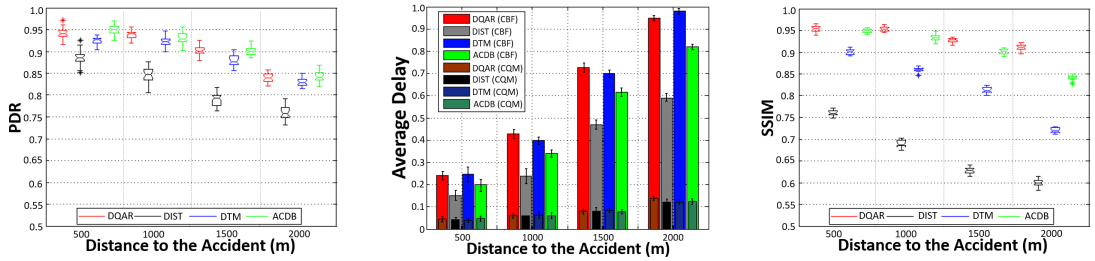
selection of FNCs for each protocol. We introduced these improvements because the standard protocols, as they were, did not represent a fair comparison.

The I-, P,- and B-frame weights (α_1, α_2, and α_3) influence the QAR performance. We have conducted independent empirical evaluations and we concluded that $\alpha_1 = 0.65$, $\alpha_2 = 0.3$, and $\alpha_3 = 0.05$ give the best Vid_{param} results. Moreover, the weights for each parameter ω_1 and ω_2 were fixed in 0.6 and 0.4, respectively, which allow QAR to achieve the best trade-off between the lowest number of hops and the enough QoE-indicators to assure the video delivery with an acceptable video quality level. In addition, we set CW_{Max} to 100ms, $W(VF)$ to 80ms, and $\overline{D_t}$ to 0.75.

We assessed the above protocols by reachability, i.e., the average fraction of nodes that receive the broadcasted videos, number of FNs over receiving nodes, Packet Delivery Rate (PDR), and average delay. Since measuring the human experience is key for our work, we carried out QoE-based measurements with a well-known objective QoE metric, called Structural SIMilarity (SSIM) [Quadros et al. 2016]. SSIM measures the structural distortion of the video to obtain a better correlation with the user's perspective. We obtained results by varying number of vehicles (50 - 200 veh/km^2) and distance to the crash area (500 - 2000m), being an average of 35 simulations (95% confidence level). Each simulation lasts 500s, where, a SN sends a VF at any time after the initial 100s and before the last 100s.

Fig. 3a shows DQAR, DIST, and DTM outperforming ACDB in terms of reachability, due to the latter unconsider positioning parameters to selection of FNs (counter-based protocol). When in 50 Veh/km^2, all protocols achieve a less reachability due to irregular mobility and distribution of vehicles in the network, causing void areas. When between 150 and 200 Veh/km^2, all the distance-based protocols perform with reachability between 93% and 99%, since for high density scenarios, the route options are greater, leading to fewer broken links and a bigger coverage of protocols. On average, DQAR and DTM increase reachability by 15.3% and 14.8% compared to ACDB, respectively, while DIST increases reachability by 4.2% and 5.3% over DQAR and DTM, respectively. The highest reachability of DIST occurs due to the selection parameters of DQAR and DTM, where FNs are close nodes that experience more stable connectivity.

Fig. 3b shows the number of FNs over receiving nodes as a measure of the protocols efficiency. The smaller the number of FNs with satisfactory reachability for a video broadcast, the more efficient is the protocol. In Fig. 3b, DQAR and DTM have similar



(a) (b) (c)

Figure 4. PDR, Average Delay, and SSIM vs Distance to the Accident

behavior, however, DIST provides a lower performance, whereas ACDB achieves good results when compared to DQAR and DTM. DQAR, DIST and DTM are identical except DIST uses only the distance-based strategy, DTM uses the Distance-to-Mean strategy, and DQAR uses distance-based strategy coupled with video-related parameters to reinforce the QoE. ACDB has an adaptive FF, which depends on the density of vehicles. Thus, the more vehicles, the lower the FF for transmission, leading to a smaller number of FNs per receiving nodes. In Fig. 3a, DQAR, DTM, and DIST exhibit a close reachability. However, in terms of number of FNs to accomplish that level of reachability, DQAR and DTM uniformly consume less bandwidth than DIST. Finally, despite these apparent good findings in terms of overhead and efficiency, ACDB does not achieves a good reachability, thus, it does not performs satisfactorily for warning video dissemination over VANETs.

Regarding the performance in terms of PDR and average delay, Figs. 4a and 4b show the performance results for the four simulated protocols for vehicles located within different distances from the accident region when the transmission of video packets was initiated. As aforementioned, DIST achieves a high reachability, but faces several broken link situations leading to a low PDR. Otherwise, ACDB reaches a PDR slightly higher in comparison to DTM, i.e., around 3.1%. This is because sometimes DTM elects farthest relay nodes such as DIST, mainly when there are few neighboring vehicles. Otherwise, ACDB adapts its FF depending on the number of neighbors, thus, increasing PDR.

The impact of the CBF and CQM phases are meaningful on the delivery delay over the transmissions (Fig. 4b), since CQM allows a great reduction of the average delay by using a contention-free forwarding. Here, we consider the average delay required by a $W(VF)$ to be transmitted in a range starting from SN. DIST at CBF stage experiences the lower delay, due to the reduced number of hops achieved by the distance-Based strategy. After, DIST is followed by ACDB, because this protocol reduces its $BackoffTimer$ when in presence of few nodes. In addition, when DIST is compared with DQAR and DTM, the delay reduction provided by DIST is significant, e.g., around 40.2% and 38.6%, respectively for 1000m of range. As aforementioned, DQAR, unlike DIST and according to its forwarding parameters, provides more effort to transmit flows with high QoE, this could mean forwarding streams to closer nodes, increasing the average delay. However, the achieved delay levels are negligible even in video applications and are significantly lower than the requirements of 4 to 5 s defined by CISCO [Index 2013].

As discussed before, QoS-based metrics (e.g., PDR) are not enough to measure the quality level from the user's perspective. Thus, aiming to understand and confirm

the impact of the video-related parameters, the results in Fig. 4c present the SSIM metric. SSIM values range from 0 to 1, where higher values mean better video quality. In Fig. 4.c, DQAR keeps the SSIM values about 0.97 and 0.9. An average increase of 27.2%, 18.1%, and 8.3% compared to DIST, DTM, and ACDB, respectively. It presents more deeply results than those obtained in Fig. 4a and shows significant benefits to the user's experience. This occurred because DQAR perceives when the QoE of the transmitting flow decreases based on the different received frame types, codec configurations, and losses, allowing eNB and vehicles, through FF calculation, to switch to others nodes, before increasing damage on the flow quality. For instance, let us suppose a $W(VF_i)$ successfully received by a VR_i in $|W(VF_i)|$ ms. As the spatial distribution of vehicles does not change very quickly in a short period of time (e.g., 3 s), it is likely that VR_i, continue to receive successfully a greater number of packets until a new route becomes necessary. Thus, DQAR provides a trade-off between hop-length and video quality. Aiming to give the reader an idea of the user's point-of-view, in [Video Sequences 2016b], we provide some simulated video samples for comparison between the four evaluated routing protocols in this paper.

5. Conclusion

In this paper, we introduced QAR to optimize warning video dissemination with QoE-awareness in LTE-VANETs. QAR aims to share videos with a better quality than existing works by applying video-related parameters to the selection of forwarding nodes and changing routes as soon as the video quality levels are bellow to a QoE-aware threshold. Results highlighted the performance and QoE-awareness of QAR by measuring the video quality levels when the distance to the sender varies. From our analysis, we identified that the distance-based protocols (i.e., DIST and DTM) perform poorly compared to DQAR in terms of QoE assurance. Despite this, these protocols have obtained similar delivery rate and average delay levels. According to its forwarding parameters, QAR provides a greater support to dissemination of video flows with higher quality from the user's point-of-view. This could mean forwarding of streams to alternative nodes by increasing transmission delays, but nonetheless, still are insignificant to the real-time video requirements. In future works, we will perform a study with other video features (e.g., GoP size and packetization), so that FF_t can be dynamically adjusted to better performance.

References

Chang, S.-w. and Lee, S.-s. (2015). Distance-based stable routing decision scheme in urban vehicular ad hoc networks. *Int. Journal of Distributed Sensor Networks*, 2015:3.

Chekhar, M., Zine, K., Bakhouya, M., and El Ouadghiri, D. (2016). An efficient broadcasting scheme in mobile ad-hoc networks. *Procedia Computer Science*, 98:117–124.

da Silva, C., Ribeiro, E., and Pedroso, C. (2016). Preventing quality degradation of video streaming using selective redundancy. *Computer Communications*, 91:120–132.

De Felice, M., Cerqueira, E., Melo, A., Gerla, M., Cuomo, F., and Baiocchi, A. (2015). A distributed beaconless routing protocol for real-time video dissemination in multimedia vanets. *Computer Communications*, 58:40–52.

Gonzalez, S. and Ramos, V. (2016). Preset delay broadcast: a protocol for fast information dissemination in vehicular ad hoc networks (vanets). *EURASIP Journal on Wireless Communications and Networking*, 2016(1):1–13.

Husain, A. and Sharma, S. (2016). Performance analysis of location and distance based routing protocols in vanet with ieee802. 11p. In *3rd International Conference on Advanced Computing, Networking and Informatics*, pages 215–221. Springer.

Index, C. V. N. (2013). Global mobile data traffic forecast update, 2012–2017, cisco white paper, feb. 6, 2013.

Jauregui, B. B. and Malaina, F. L. (2016). new approaches to mobile ad hoc network routing: Application of intelligent optimization techniques to multicriteria routing. *Mobile Ad Hoc Networks: Current Status and Future Trends*, page 171.

Jia, S., Hao, S., Gu, X., and Zhang, L. (2014). Analyzing and relieving the impact of fcd traffic in lte-vanet heterogeneous network. In *Telecommunications (ICT), 2014 21st International Conference on*, pages 88–92. IEEE.

Mir, Z. H., Kim, J., Ko, Y.-B., and Filali, F. (2016). Improved multi-hop routing in integrated vanet-lte hybrid vehicular networks. In *10th International Conference on Ubiquitous Information Management and Communication*, page 76. ACM.

Quadros, C., Santos, A., Gerla, M., and Cerqueira, E. (2016). Qoe-driven dissemination of real-time videos over vehicular networks. *Computer Communications*, 91:133–147.

Salvo, P., Turcanu, I., Cuomo, F., and Baiocchi, A. (2016). Lte floating car data application off-loading via vanet driven clustering formation. In *12th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pages 1–8. IEEE.

Sherry, J., Lan, C., Popa, R. A., and Ratnasamy, S. (2015). Blindbox: Deep packet inspection over encrypted traffic. In *ACM SIGCOMM Computer Communication Review*, volume 45, pages 213–226. ACM.

Slavik, M., Mahgoub, I., and Alwakeel, M. (2015). Efficient multi-hop wireless broadcast protocol in vehicular networks using automated threshold function design. *International Journal of Communication Systems*, 28(12):1829–1846.

Tao, S., Apostolopoulos, J., and Guérin, R. (2008). Real-time monitoring of video quality in ip networks. *IEEE/ACM Transactions on Networking (TON)*, 16(5):1052–1065.

Torres, A., Calafate, C. T., Cano, J.-C., and Manzoni, P. (2015). Evaluation of flooding schemes for real-time video transmission in vanets. *Ad Hoc Networks*, 24:3–20.

Ucar, S., Ergen, S. C., and Ozkasap, O. (2016). Multihop-cluster-based ieee 802.11 p and lte hybrid architecture for vanet safety message dissemination. *IEEE Transactions on Vehicular Technology*, 65(4):2621–2636.

Video Sequences, N. R. L. (2016a). Videos used in the simulations ('truck accident') < URL https://www.youtube.com/channel/UCUlJSLvBpeJbLAr6IsVXbJA/videos>.

Video Sequences, U. F. P. A. (2016b). Videos used in the QoE measurements ('truck accident') < URL http://www.gercom.ufpa.br/sbrc2017-videoexperiments>.

Zaimi, I., Houssaini, Z. S., Boushaba, A., Oumsis, M., and Aboutajdine, D. (2016). Vehicular ad-hoc network: Evaluation of qos and qoe for multimedia application. In *International Conference on Networked Systems*, pages 367–371. Springer.

GTE: Um Sistema para Gerenciamento de Trânsito Escalável baseado em Compartilhamento Oportunista

Allan M. de Souza[1], **Leonardo C. Botega**[2], **Leandro A. Villas**[1]

[1]Instituto de Computação – UNICAMP
allanms@lrc.ic.unicamp.br, leandro@ic.unicamp.br

[2]Centro Universitário Eurípides de Marília – UNIVEM
botega@univem.edu.br

Abstract. *Urban mobility has become one of the most challenging issue in large urban centers. As a consequence, traffic congestion has become a daily problem. Several Traffic Management Systems (TMS) have been proposed to improve overall traffic efficiency. However, the proposals exchange inefficiently traffic information, which can lead to network overload. In order to overcome the mobility problem and improve the efficiency in dealing with vehicle traffic, this work introduces a scalable traffic management system based on opportunistic content sharing, named GTE. Simulation results indicate that GTE outperforms the assessed solutions in different scenarios and in different key requirements of TMS.*

Resumo. *Mobilidade em grandes centros urbanos tornou-se um grande desafio, como consequência dos congestionamentos recorrentes. Dessa forma, vários* Traffic Management Systems *(TMS) têm sido propostos para melhorar a eficiência do tráfego. Entretanto, muitas das soluções propostas não são escaláveis, utilizando-se de métodos limitados para troca de informações, o que pode sobrecarregar a rede. Além disso, outras soluções não são capazes de fornecer um conhecimento preciso sobre as condições de tráfego. Dessa forma, para minimizar o problema de mobilidade e melhorar a eficiência do tráfego, introduzimos GTE - Gererenciamento de Tráfego Escalável, um TMS totalmente distribuído e escalável baseado em compartilhamento oportunista. Os resultados mostram que o GTE supera as soluções avaliadas em diferentes requisitos essenciais de um TMS.*

1. Introdução

Congestionamento tornou-se um problema diário em grandes centros urbanos, resultando-se do grande crescimento populacional juntamente com o aumento do número de veículos, excedendo a capacidade da infraestrutura de transporte [de Souza et al. 2016, Djahel et al. 2015]. Diante desse cenário, outros problemas podem emergir, incluindo degradação da qualidade de vida, impactos negativos na economia regional e danos ambientais [Karagiannis et al. 2011, de Souza et al. 2016].

A fim de maximizar a eficiência da infraestrutura de transporte disponível, minimizando o congestionamento e seus danos, Sistemas de Gerenciamento de Trânsito (*TMS – Traffic Management System*) foram propostos. TMSs concentram-se na integração de tecnologias de sensoriamento, comunicação e processamento de dados para coletar e explorar dados relacionados ao trânsito, fornecidos por veículos, sensores, subsistemas e até pessoas, com o objetivo de identificar eventos que podem degradar a eficiência do trânsito e prover serviços para minimizar os efeitos desses eventos [Djahel et al. 2015].

A base para a implantação de um TMS eficiente é a Rede Veicular, a qual é uma rede *ad hoc* móvel composta por veículos equipados com sensores, unidades de processamento e interfaces de comunicação sem fio [Karagiannis et al. 2011]. Devido a essas características,

os veículos podem comunicar entre si criando uma rede *ad hoc*, ou eles podem se comunicar com *Roadside Units* (RSUs), as quais são entidades com capacidade de processamento, sensoriamento e comunicação localizadas próximo à infraestrutura de transporte e são utilizadas para maximizar a capacidade da rede e seu gerenciamento, bem como prover acesso a Internet. Em suma, as Redes Veiculares permitem a troca de informações entre os veículos e infraestrutura, e também fornecem uma plataforma para sensoriamento e atuação para os TMSs [Djahel et al. 2015].

Diferentes TMSs foram propostos para lidar com as necessidades de mobilidade urbana e melhorar a eficiência do tráfego [Araujo et al. 2014, de Souza et al. 2015, Wang et al. 2015, Pan et al. 2016, Doolan and Muntean 2016]. Algumas soluções [de Souza et al. 2015, Wang et al. 2015] utilizam uma abordagem centralizada para detectar iminência de um congestionamento como também para sugerir rotas alternativas. Entretanto, essas soluções não são escaláveis e podem apresentar um *overhead* indesejado, especialmente em cenários de alta densidade de veículos, como por exemplo as altas densidades de veículos durante horários de pico [de Souza et al. 2015, Wang et al. 2015, Pan et al. 2016, Doolan and Muntean 2016]. Por outro lado, outras soluções como [Araujo et al. 2014] utilizam uma abordagem distribuída, porém, não possuem conhecimento da real condição de tráfego corrente, o que pode incorrer em orientações de rotas errôneas.

Dessa forma, devido às limitações apresentadas pelos TMSs atuais, tais como, falta de escalabilidade, *overhead* introduzido em altas densidades, e a falta de conhecimento sobre as reais condições de tráfego, torna-se desejável uma abordagem escalável, a qual lida com esses problemas. Entretanto, um desafio chave nesta abordagem é como fornecer um conhecimento preciso sobre a real condição de tráfego corrente para todos os veículos em circulação, um vez que soluções simples podem potencialmente sobrecarregar a rede e introduzirem um *overhead* indesejado.

Visando superar o desafio supracitado, propomos GTE (Gererenciamento de Tráfego Escalável), um TMS distribuído e escalável projetado para maximizar a eficiência do tráfego sem sobrecarregar a rede. GTE concentra-se em fornecer um conhecimento preciso sobre as condições de tráfego em áreas críticas (áreas com congestionamentos recorrentes) em vez de fornecer conhecimento geral sobre as condições de tráfego. Além disso, GTE utiliza uma abordagem de *Conteúdo Flutuante*, a qual atua como um serviço baseado em localização para disseminar informações geo-localizadas ao invés de disseminar a informação para toda a rede [Castro et al. 2008, Hyytiä et al. 2011]. De forma geral, um conteúdo flutuante é associado a uma área geográfica, onde o mesmo é responsável por se manter disponível para os veículos localizados nesta área, bem como para os veículos que irão adentrar na mesma, utilizando um compartilhamento de informação oportunista. Portanto, ao aproximar-se de uma área crítica ou adentrá-la, os veículos receberão informações precisas sobre a condição de tráfego corrente, permitindo-lhes detectar vias congestionadas e calcular rotas alternativas para evitá-las.

Neste trabalho, é apresentada uma série extensiva de experimentos que mostram a necessidade de um TMS escalável para maximizar a eficiência do trânsito. Adicionalmente, resultados dos experimentos com GTE indiciam um desempenho superior quando comparado a soluções existente.

O restante do trabalho encontra-se organizado da seguinte forma. A Seção 2 fornece uma visão geral dos TMS existentes e apresenta alguns trabalhos relacionados. A Seção 3 descreve a solução proposta. A análise de desempenho da solução proposta é apresentada na Seção 4. Por fim, a Seção 5 apresenta as conclusões e indica trabalhos futuros.

2. Trabalhos Relacionados

Os TMSs atuais podem ser amplamente classificados em *(i)* Centralizados; *(ii)* Distribuídos; e *(iii)* Híbridos. TMSs centralizados têm o propósito de fornecer um melhor gerenciamento do tráfego, utilizando uma abordagem centralizada para detectar congestionamentos e sugerir rotas alternativas [Wang et al. 2015, de Souza et al. 2015]. Para isso, estas soluções baseiam-se em informações reportadas por veículos periodicamente como posição, velocidade média, destino, etc. TMSs distribuídos [Araujo et al. 2014] utilizam uma abordagem cooperativa para detectar congestionamentos e evitá-los baseado em informações compartilhadas entre os veículos. TMSs híbridos [Doolan and Muntean 2016, Pan et al. 2016] utilizam uma abordagem centralizada para detectar congestionamentos e construir um conhecimento sobre a condição de tráfego, porém, em seguida utilizam uma abordagem distribuída para calcular rotas alternativas, onde o conhecimento é encaminhado para os veículos, para que os mesmos possam calcular novas rotas.

Contudo, TMSs centralizados não são escaláveis, pois podem sobrecarregar a rede em determinadas densidades de veículos e introduzir um *overhead* indesejado [Wang et al. 2015, de Souza et al. 2015, Doolan and Muntean 2016, Pan et al. 2016]. TMSs híbridos, por utilizarem uma entidade central para agregar o conhecinmento do tráfego, também pode não ser escalável, e assim, introduzir uma latência indesejada para o sistema se troca de informações não for feita de maneira eficiente. Por fim, TMSs distribuídos [Araujo et al. 2014] apesar de apresentarem uma alta escalabilidade, podem incorrer no mesmo problema. Adicionalmente, eles não possuem um conhecimento preciso da condição de tráfego corrente [Araujo et al. 2014].

Araújo et al. [Araujo et al. 2014], propuseram CARTIM, um TMS distribuído para minimizar congestionamentos. Os veículos mensuram os níveis de congestionamento em suas vias baseado informações compartilhadas por mensagens *beacons* enviadas por todos os veículos. Dessa forma, baseado nas informações recebidas e utilizando um sistema baseado em lógica fuzzy, cada veículo é capaz de classificar o nível de congestionamento em sua via. Posteriormente, os veículos compartilham suas classificações com seus vizinhos, para cooperativamente detectar um congestionamento. Além disso, quando uma via congestionada é detectada, os veículos calculam uma rota alternativa utilizando uma heurística que baseia-se na distância até o destino. Entretanto, essa heurística pode incorrer em uma má orientação de rota, uma vez que os veículos não possuem conhecimento sobre a condição de tráfego nas vias.

Doolan and Muntean [Doolan and Muntean 2016] introduziram EcoTrec, um TMS híbrido que baseia-se no roteamento periódico dos veículos. Por ser uma solução híbrida, EcoTrec utiliza um servidor central para coletar informações dos veículos e construir um conhecimento sobre a condição de tráfego do cenário todo, em seguida, esse conhecimento é disseminado para os veículos para que eles possam calcular uma rota baseada no menor consumo de combustível. Para enviar informações para o servidor, EcoTrec utiliza um protocolo de disseminação, o qual não implementa nenhum mecanismo de supressão de *broadcast*, onde, por sua vez pode introduzir um *overhead* para o sistema em cenários densos.

Em nosso trabalho anterior [de Souza et al. 2015], propomos CO-OP, um TMS centralizado que classifica a condição de tráfego no cenário todo baseado no algoritmo *K-Nearest Neighbor* (KNN). Igualmente o EcoTrec, no CO-OP todos os veículos precisam enviar suas informações para um servidor central, para que o mesmo possa detectar congestionamentos e sugerir rotas alternativas. Quando um congestionamento é detectado, CO-OP emprega um roteamento cooperativo, o qual sugere rotas alternativas para os veículos que vão passar pelo

congestionamento detectado. Este roteamento cooperativo aplica um balanceamento de carga para distribuir melhor o fluxo de veículos em todo o cenário. Entretanto, como EcoTrec, CO-OP por introduzir um *overhead* indesejado em altas densidades de veículos.

Os trabalhos supradescritos apresentam algumas limitações como baixa escalabilidade, dependência de uma RSU ou servidor central, sobrecarga da rede [de Souza et al. 2015, Doolan and Muntean 2016, Wang et al. 2015, Pan et al. 2016], e a ausência de conhecimento sobre a real condição de tráfego para uma melhor orientação de rotas [Araujo et al. 2014]. Neste contexto, para superar tais limitações apresentadas, propomos um TMS distribuído baseado em compartilhamento oportunista, o qual não sobrecarrega a rede e também não introduz uma latência indesejada. Além disso, o TMS proposto fornece um conhecimento preciso sobre a condição de tráfego.

3. Gerenciamento de Trânsito Escalável

GTE é um TMS baseado em conteúdo flutuante, proposto para melhorar a eficiência do tráfego. Este emprega uma abordagem de compartilhamento oportunista para fornecer e armazenar informações sobre as reais condições de tráfego em regiões críticas. Para isso, GTE não depende de nenhuma RSU ou entidade central, consequentemente os veículos são aptos a detectar vias congestionadas e calcular uma rota mais eficiente.

Definição: *Considere um ambiente veicular representado por um grafo simples e ponderado* $G = (V(G), E(G))$*, onde o conjunto* $V = \{v_1, v_2, \ldots, v_i\}$ *representa o conjunto de interseções do cenário (vértices), o conjunto* $E = \{e_{01}, e_{12} \ldots, e_{ij}\}$ *representa o conjunto de ruas do cenário (arestas) as quais conectam as interseções* $E(G) \subseteq V(G) \times V(G)$*, onde a aresta* e_{ij} *é definida por um par de vértices subsequentes* $(v_i, v_j) \in V(G)$*. Além disso,* $W = \{w_{01}, w_{12}, \ldots, w_{ij}\}$ *é o conjunto de pesos que representa a condição de tráfego em cada rua do cenário, na qual* $w : E \to \mathbb{R}^*_+$*. Seja* $N = \{n_1, n_2, \ldots, n_i\}$ *o conjunto de veículos em circulação e* $R = \langle e_{ij}, \ldots, e_{mn} \rangle \mid R \subseteq E(G)$ *uma rota* $\forall\, n \in N$*.*

Um desafio chave em prover um TMS distribuído, é como construir e entregar para todos os veículos um conhecimento preciso sobre a condição de tráfego do cenário, sem sobrecarregar a rede e consequentemente produzindo um baixo *overhead*. Pois, as informações geradas por cada veículo precisam ser entregues para todos os outros veículos na rede, portanto, caso este processo não for feito de forma eficiente, muitas transmissões em um breve intervalo podem sobrecarregar a rede, produzindo um *overhead* indesejado e degradar a eficiência do sistema [de Souza et al. 2016]. Consequentemente, os veículos não terão um conhecimento preciso sobre a condição de tráfego, incapacitando-os de detectar vias congestionadas. Sendo assim, para lidar com esse problema, GTE constrói um conhecimento preciso sobre as reais condições de tráfego para em áreas críticas, ou seja, áreas conhecidas por possuir congestionamentos recorrentes ou alto número de incidentes de tráfego. Além disso, GTE usa uma abordagem de compartilhamento oportunista para prover esse conhecimento como um conteúdo flutuante por toda área crítica. Portanto, sempre que um veículo entrar em uma área crítica, o mesmo receberá o conteúdo flutuante, permitindo-o detectar vias congestionadas em sua rota corrente.

Áreas críticas possuem uma localização (x_c, y_c) e uma área $C = (h \times l)$, os quais definem seu centro (ponto crítico) e a área potencialmente afetada por incidentes dentro da mesma. Para cada conteúdo flutuante de cada área crítica, GTE atribui uma *Área de Ancoragem* (AA) definida por um raio $r_a \in \mathbb{R}^*_+ \mid r_a \geq \frac{h}{2} \geq \frac{l}{2}$, o qual é baseado no ponto crítico (x_c, y_c). Cada AA é responsável por manter o conteúdo flutuante dentro dela, tornando-o disponível para os veículos que estão localizados dentro da mesma e para os outros veículos que irão passar por ela. Essas AAs podem ter tamanhos diferentes e podem ser utilizadas para

diferentes propósitos [Hyytiä et al. 2011]. Além disso, elas podem ter locais conhecidos ou dinâmicos, ou seja, elas podem surgir assim que um incidente de trânsito ocorre, a fim de fornecer conhecimento sobre o incidente para os veículos que estão próximos do incidente, assim como para os veículos que irão passar por essa área.

A Figura 1 apresenta as AAs para dois conteúdos flutuantes. Particularmente, AAs (AA_1 and AA_2) possuem tamanhos diferentes e seus respectivos conteúdos armazenados. Os veículos n_1, n_2 e n_3 possuem o conteúdo flutuante fornecido pela AA_1, enquanto os veículos n_4, n_5, n_6 e n_7 possuem o conteúdo flutuante fornecido pela AA_2. É importante notar que, se um veículo estiver em localizado em um local onde tem interseção de duas ou mais AAs, ele potencialmente pode possuir mais de um conteúdo flutuante, neste caso o veículo n_8 possui os conteúdos flutuante fornecidos pelas duas AAs. Apesar das áreas críticas conhecidas

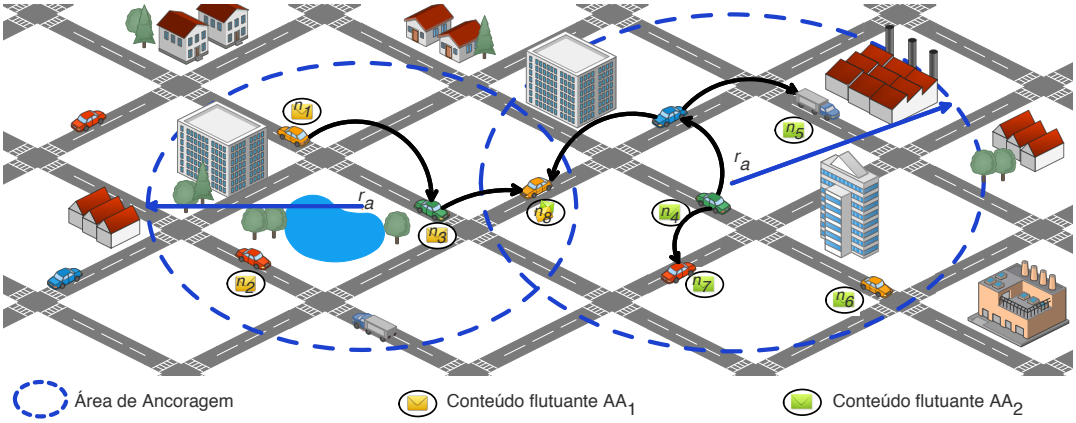


Figura 1. Conteúdos flutuantes e suas respectivas áreas de ancoragem.

pelo GTE, o mesmo não possui nenhum conhecimento sobre as condições de tráfego dentro delas. Neste caso, GTE emprega um mecanismo eficiente para suprir essa necessidade, construindo um conhecimento preciso sobre as condições de tráfego dentro dessas áreas, os quais, em seguida são disponibilizados como conteúdo flutuante, garantindo um compartilhamento oportunista e a sobrevivência do mesmo nesta área. Por fim, ele também fornece um mecanismo para cada veículo detectar vias congestionadas, permitindo-os a ter uma melhor orientação de rota. Sendo assim, a Subseção 3.1 descreve como este conteúdo flutuante é criado, Subseção 3.2 descreve como GTE garante a disponibilidade e sobrevivência de cada conteúdo flutuante, e Subseção 3.3 descreve como os veículos detectam vias congestionadas e computam rotas alternativas para si mesmos para evitar vias congestionadas.

3.1. Criação do conteúdo flutuante

GTE baseia-se em informações periódicas compartilhadas por todos os veículos através de mensagens *beacons*, as quais possuem a posição atual do veículo $e_{ij} \in R$, velocidade média s_m, direção e um identificador para informar se o veículo armazena algum conteúdo flutuante e se o mesmo está atualizado.

Dado uma área crítica, GTE define um conjunto de *veículos produtores* $N' \subseteq N$, os quais estão localizados dentro da área crítica e são responsáveis por construir o conhecimento sobre a condição de tráfego naquela área. Entretanto, existem alguns desafios a serem considerados: *(i)* como garantir a cobertura total de uma área crítica, uma vez que o raio de comunicação pode ser muito menor do que a área crítica, e *(ii)* como construir o conhecimento sem sobrecarregar a rede.

Neste caso, GTE segmenta toda área crítica em k regiões menores chamadas de *subáreas*. para primeiro construir um conhecimento sobre o tráfego de cada subárea, e em seguida agregar o conhecimento de cada subárea em um único conhecimento, o qual representa a condição de tráfego de toda área crítica. A quantidade de subáreas e definida baseada no tamanho do raio de comunicação dos veículos [de Souza and Villas 2016].

Antes de construir o conhecimento de cada subárea, primeiramente cada veículo produtor $n_i \in N'$ cria um conhecimento local sobre o conhecimento do tráfego agregando as mensagens de *beacons* de seus vizinhos. Este conhecimento local é representado pelo conjunto $E^{n_i} = \{e_{ij,\ldots,e_{mn}}\} \mid E^{n_i} \subset E(G)$, o qual é composto pelas vias dentro do raio de comunicação do veículo produtor e suas respectivas velocidades médias. Após criar seu conhecimento local E^{n_i}, cada veículo produtor compartilha-o com seus vizinhos localizados na mesma subárea, a fim de criar um conhecimento sobre toda a subárea, a função de agregação é dada por:

$$S_i = \frac{1}{n} \sum_{n_i \in N'} e_{ij}, \forall e_{ij} \in E^{n_i} \tag{1}$$

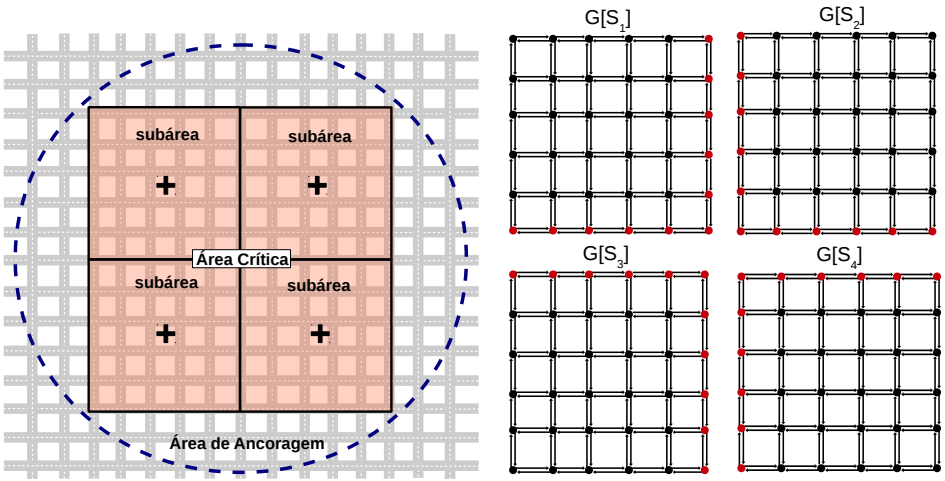
onde, S_i com $i \in \{1, 2, \ldots, k\}$ representa o conhecimento sobre a condição de tráfego de cada subárea formado pela agregação do conhecimento E^{n_i} recebido de todos os veículos produtores dentro dela. Além disso, baseado na velocidade média de cada via, cada veículo atribui um peso para a mesma utilizando a seguinte fórmula:

$$w_{ij} = 1 - \max(e_{ij})^{-1} \times e_{ij} \;\; \forall e_{ij} \in E^{n_i} \tag{2}$$

onde $\max(e_{ij})$ e e_{ij} representam a velocidade máxima e média de cada via.

Posteriormente, todo veículo localizado na mesma subárea cria um subgrafo $G[S_i] \subset G$ induzido pelas arestas com o conhecimento agregado S_i, onde o conjunto $E(G[S_i]) \subset E(G)$ representa o conjunto de vias pertencentes a cada subárea, enquanto o conjunto $V(G[S_i]) \subset V(G)$ representa o conjunto de interseções que conectam as vias da subárea $e_{ij} \in E(G[S_i])$. Portanto, para criar o conteúdo flutuante com o conhecimento sobre o tráfego de toda área crítica, é necessário integrar cada subgrafo $G[S_i]$ induzido pelas arestas de cada subárea em um único supergrafo G'. Sendo assim, cada subgrafo de cada subárea precisa ser encaminhado para todas as outras subáreas. Para isso, GTE emprega um protocolo de disseminação de dados entre as subáreas baseado em [de Souza et al. 2016], tal protocolo utiliza um mecanismo de supressão de *broadcast* e outro para minimizar a quantidade de colisões de pacotes evitando a ressincronização introduzida pelo padrão IEEE 802.11p utilizado em Redes Veiculares.

Graças à segmentação da área crítica juntamente com o protocolo de disseminação de dados para agregar o conhecimento de todas as subáreas, GTE maximiza a cobertura reduzindo o número de transmissões, e ainda, garante pelo menos uma transmissão dentro de cada subárea. Para isso, GTE seleciona um veículo de cada subárea para iniciar o processo de disseminação, este veículo é selecionado baseado em sua posição corrente e também com base em seu último conhecimento gerado, para saber se o mesmo encontra-se atualizado. Portanto, todos os veículos na mesma subárea agendam uma transmissão de uma mensagem que contém o conhecimento agregado da subárea com um identificador da subárea em questão, cada transmissão é agendada baseada na distância euclidiana em que os veículos encontram-se do centro de sua subárea, assim, permitindo que veículos mais próximos do centro transmitam primeiro, e devido ao mecanismo de supressão de *broadcast* implementado, quando veículos que haviam agendado a transmissão da mesma mensagem recebem uma mensagem com o mesmo identificador, os mesmos cancelam a transmissão, evitando a transmissão de mensa-



(a) Áreas de ancoragem, área crítica, k-subáreas e seus respectivos centros (b) Subgrafo induzido pelas arestas de cada subárea.

Figura 2. Área crítica, subáreas, AA e subgrafo induzido pelas arestas da subárea.

gens redundantes. Note que, como todas as subáreas são distintas, as transmissões de uma subárea não cancelam transmissões de outra subárea.

Entretanto, se tal segmentação não fosse utilizada, essa mesma abordagem de disseminação geraria lacunas no conhecimento geral da área crítica, uma vez que a transmissão de um determinado veículo poderia potencialmente cancelar transmissões de outros veículos com informações diferentes, onde essas informações seriam importantes para fornecer um conhecimento preciso sobre a condição de tráfego em toda área crítica [de Souza et al. 2016]. Além disso, o processo de encaminhamento empregado pelo GTE não sobrecarrega a rede [de Souza et al. 2016].

Depois de receber todos os conhecimentos (subgrafos $G[S_i]$) de cada subárea, GTE cria o conteúdo flutuante o qual contém um subgrafo $G' \subseteq G$ dado pela Equação 3, a qual representa toda a área crítica, contendo o conhecimento preciso sobre a condição de tráfego, um raio de ancoragem r_a para determinar sua geolocalização, uma variável temporal para informar quando o conteúdo foi criado e um identificador para identificar a qual área crítica o mesmo pertence.

$$G' = \bigcup_{i=0}^{k} G[S_i] \tag{3}$$

Por questão de clareza, a Figura 2 descreve cada subárea de uma área crítica com o subgrafo G' criado pelo GTE. Em particular, a Figura 2(a) apresenta a segmentação implementada pelo GTE apresentando a área de ancoragem, área crítica, k-subáreas e seus respectivos centros. Adicionalmente, a Figura 2(b) apresenta cada subgrafo $G[S_i]$ induzido pelas arestas de cada subárea. Os pontos vermelhos indicam os vértices em comum em dois ou mais subgrafos, estes vértices são utilizados para fazer a união dos subgrafos $G[S_i]$ em um supergrafo G'. É importante salientar que este processo de união dos subgrafos resulta em um grafo simples, ou seja, as arestas múltiplas geradas no processo de união são removidas.

3.2. Compartilhamento oportunista

O compartilhamento oportunista é essencial para garantir a sobrevivência de um determinado conteúdo flutuante em sua AA. Ele mantém cada conteúdo flutuante em sua AA, tornando-o

disponível para os veículos dentro da área crítica, e também para os veículos que irão chegar nesta área. Além da AA, GTE implementa uma outra área, nomeada *Área de Encaminhamento* (AE), a qual é definida por $r_f \in R_+^* \mid r_f > r_a$ e é responsável por enviar de volta o conteúdo flutuante armazenado em veículos que estão deixando a AA. Em outras palavras, veículos que saíram da AA mas ainda possuem o conteúdo flutuante, precisam procurar ativamente por veículos que estão indo em direção da AA para encaminhar o conteúdo flutuante.

A fim de evitar transmissões desnecessárias, quando um veículo que armazena um conteúdo flutuante deixa sua AE, ele descarta o conteúdo para evitar o encaminhamento para veículos que estão muito distantes, uma vez que eles não possuem interesse naquele conteúdo neste momento, e a condição de tráfego de tal área crítica possivelmente pode mudar até o veículo chegar nessa área. Além disso, assim que os veículos se aproximaram de uma área crítica, os mesmos irão receber um conteúdo flutuante pertencente a esta área com um conhecimento atualizado sobre a condição de tráfego.

O processo de encaminhamento garante a sobrevivência do conteúdo em sua AA e o mesmo é baseado no interesse que o veículo possui em tal conteúdo. Resumidamente, o interesse depende se o veículo vai passar por uma área crítica, e se o mesmo possui um conhecimento atualizado sobre o local. Estas informações são obtidas através das mensagens de *beacons* periodicamente compartilhadas por todos os veículos. Portanto, quando um veículo que armazena um conteúdo flutuante atualizado recebe uma mensagem de *beacon* de um outro veículo, e eles encontram-se dentro de uma das áreas AA ou AE, o mesmo irá encaminhar o conteúdo baseado em uma probabilidade p

$$p = \begin{cases} 1 & \text{se } \exists e_{ij} \in R \mid e_{ij} \in E(G') \\ \theta(R) & \text{Caso contrário} \end{cases} \tag{4}$$

onde, $\theta(R) \in [0, 1]$ é uma função decrescente que fornece a probabilidade de encaminhamento quando veículos não irão passar pela AA.

3.3. Detecção de vias congestionadas e orientação de rota

O conteúdo flutuante disponível, permite que veículos que o possuem detectem vias congestionadas e melhorem sua rota, calculando rotas alternativas para evitar as vias congestionadas detectadas. Cada veículo pode detectar vias congestionadas e classificar o nível de congestionamento baseado no peso de cada via w_{ij}, $\forall e_{ij} \in E(G')$. A classificação é baseada no *Level-Of-Service* (LOS) do *Highway Capacity Manual* (HCM)[1]. Tal manual apresenta medições de qualidade utilizadas para descrever a porcentagem de condições operacionais dentro de um fluxo de veículos, definindo seis níveis de serviços diferentes, onde LOS A representa a ausência de congestionamento e LOS F representa um alto índice de congestionamento.A Tabela 1 apresenta a relação entre o peso das vias w_{ij} e a classificação do tráfego baseado no LOS do HCM.

Como o conteúdo flutuante apenas fornece conhecimento sobre a condição de tráfego nas áreas críticas, os veículos conseguem apenas detectar vias congestionadas e calcular rotas alternativas para vias dentro da área crítica. Portanto, cada veículo verifica apenas um subconjunto de vias de sua rota completa R contido no conhecimento fornecido pelo conteúdo flutuante. Seja $R' = \{e_{ij}, \ldots, e_{xy}\} \mid R' \subseteq R \cap E(G')$ este subconjunto. Primeiramente, um veículo precisa verificar se sua rota corrente vai passar por alguma via congestionada, verificando se a via $e_{ij} \in R'$ possui um peso maior que um *threshold* ε, o qual indica a existência

[1] `http://hcm.trb.org`

Tabela 1. Relação entre o peso da via e sua classificação

Peso da via	LOS	Classificação
$(0, 0.15]$	A	Tráfego livre
$(0.15, 0.33]$	B	Tráfego livre
$(0.33, 0.50]$	C	Congestionamento Leve
$(0.50, 0.60]$	D	Congestionamento Leve
$(0.60, 0.70]$	E	Congestionada
$(0.70, 1.00]$	F	Congestionada

de algum congestionamento. Esta verificação é dada pela função $\sigma(n)$

$$\sigma(n) = \begin{cases} 1 & \text{se } \exists e_{ij} \in R' \mid w_{ij} \geq \varepsilon \\ 0 & \text{Caso contrário} \end{cases} \tag{5}$$

onde, o valor 1 significa que um veículo possui a via e_{ij} em sua rota R' que possui peso w_{ij} maior que o *threshold* ε definido, e 0 caso contrário.

Sempre que um veículo vai passar por uma via congestionada, uma rota alternativa deve ser calculada. Portanto, o veículo calcula esta rota alternativa baseado na sua posição corrente e a última via $e_{ij} \in R'$ utilizando um algoritmo de k-menores caminhos com escolha probabilística (PkSP – *Probabilistic k Shortest Paths*) baseado no peso das vias. PkSP utiliza a distribuição de Boltzamann [Kirkpatrick et al. 1983] para selecionar uma rota entre um conjunto de k-rotas calculadas, reduzindo assim a possibilidade de criar um congestionamento em outra área em um futuro próximo. Sendo assim, é importante notar que a rota alternativa completa é dada por

$$R_{alternativa} = pksp(e_{ij}, e_{xy}) \cup R' \setminus R \tag{6}$$

onde, $pksp(e_{ij}, e_{xy})$ é a rota alternativa calculada pelo veículo de sua posição atual até a última via $e_{ij} \in R'$ que o mesmo possui conhecimento sobre o tráfego e $R' \setminus R$ é o restante da rota original.

4. Análise dos resultados

Esta seção descreve a análise de performance do GTE. A Subseção 4.1 introduz as ferramentas utilizadas na simulação. As Subseções 4.2 e 4.3 apresentam a avaliação de escalabilidade do sistema e a eficiência do tráfego respectivamente. A análise de escalabilidade foi feita baseada no custo da rede avaliando o mecanismo segmentação e agregação implementado pelo GTE juntamente com a abordagem de compartilhamento oportunista. Além disso, tanto para a análise de escalabilidade, quanto para a análise de eficiência do tráfego, comparamos GTE com outros TMSs existentes na literatura.

4.1. Metodologia

Para as simulações, utilizamos o simulador de rede OMNeT++ 4.3[2] e o *framework* SUMO (*Simulator for UrbanMObility*)[3], version 0.25.0, para gerenciar o cenário e a mobilidade dos veículos. Para a rede veicular, utilizamos o *framework* Veins 4.3[4] o qual implementa o padrão IEEE 802.11p e atenuação de sinal utilizando obstáculos.

Para o cenário de simulação, utilizamos uma região de 25 km^2 da cidade de Colônia, Alemanha, obtido do projeto TAPASCologne[5] o qual tem o objetivo de reproduzir as reais

[2] `http://omnetpp.org/`
[3] `http://http://www.dlr.de/`
[4] `http://veins.car2x.org/`
[5] `http://kolntrace.project.citi-lab.fr/`

condições de tráfego da cidade em questão. Além disso, para garantir a ocorrência de um congestionamento foi utilizado apenas o intervalo das $11{:}00 - 13{:}00$, onde sabemos que há vários congestionamento na área selecionada. A partir dessas informações, definimos uma área crítica de 9 km^2 como pode ser observado na Figura 4.1, onde é apresentado o cenário utilizado juntamente com a condição de tráfego durante o horário selecionado e as regiões de 25 km^2 e 9 km^2. Por fim, a Tabela 2 descreve os principais parâmetros utilizados em nossa simulação.

Tabela 2. Parâmetros

Parâmetros	Valores
Potencia de transmissão	2.2 mW
Raio de comunicação	300 m
Bit rate	18 Mbit/s
Cenário size	25 km^2
Cenário	Colônia, Alemanha
# veículos	$\approx$ 20.000
# Áreas críticas	1
C	9 km^2
r_a	2 km
r_f	2.5 km
ε	0.5

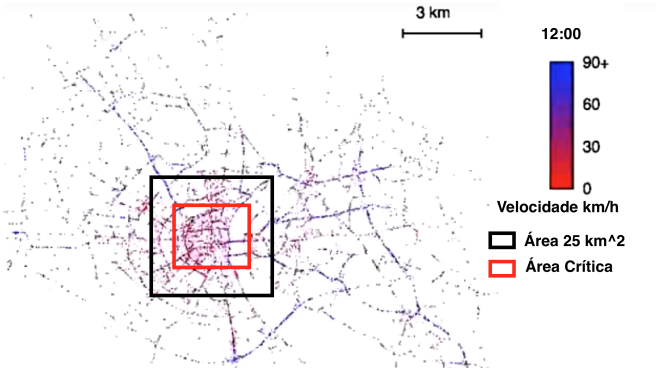


Figura 3. Colônia Alemanha.

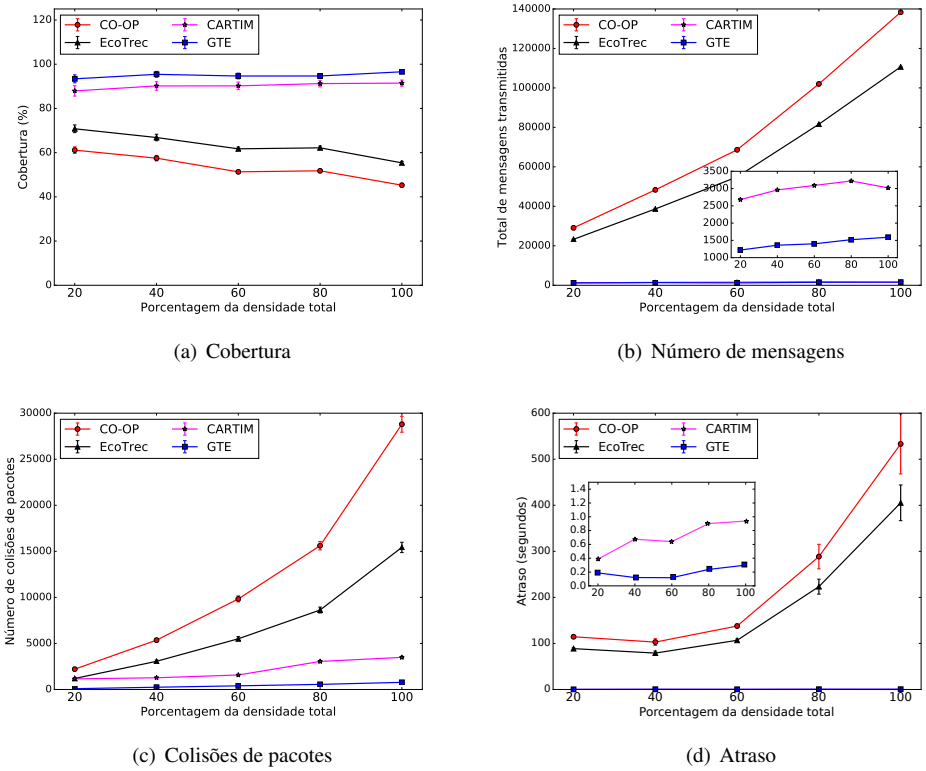
4.2. Análise de escalabilidade

Com o foco de analisar a escalabilidade dos TMSs, realizamos simulações variando a densidade de veículos de $20\% - 100\%$ da densidade original da área de 25 km^2 utilizada do cenário. Sendo assim, as métricas de avaliação são: *(i)* Cobertura: porcentagem de mensagens entregue para os veículos dentro da área crítica; *(ii)* Mensagens transmitidas: total de mensagens transmitidas para realizar a entrega do conteúdo; *(iii)* Colisões de pacote: número total de colisões de pacotes durante a transmissão do conteúdo; *(iv)* Atraso: tempo médio gasto para realizar a entrega das mensagens.

A Figura 4 apresenta os resultados para as métricas avaliadas em função da porcentagem de veículos. Em particular, a Figura 4(a) apresenta os resultados de cobertura. O CO-OP, por utilizar uma abordagem centralizada não escalável, possui menor cobertura para todas as porcentagens de densidade, atingindo uma cobertura de 60% com 20% da densidade total e aproximadamente 40% com 100% da densidade total.

Tais resultados são consequência da sobrecarga da rede (veja Figuras 4(b) e 4(c)), pois além de cada veículo ter de enviar suas informações para um servidor central, para cada veículo que vai passar por um congestionamento detectado, o CO-OP deve sugerir uma rota alternativa, aumentando drasticamente o número de mensagens transmitidas (veja Figura 4(b)). Além disso, CO-OP não implementa nenhum mecanismo de agregação ou supressão de *broadcast* para reduzir o número de mensagens transmitidas, então essa grande quantidade de mensagens transmitidas pelos veículos e servidor geram muitas colisões (veja Figura 4(c)), o que consequentemente geram outras transmissões, sobrecarregando ainda mais a rede. Essa sobrecarga e não escalabilidade do CO-OP pode ser vista na Figura 4(d), pois é introduzido um alto *overhead*, onde o atraso médio de entrega de mensagens com 100% da densidade original é de aproximadamente 10 minutos.

De outra forma, EcoTrec apresenta uma abordagem híbrida, onde, apesar dos veículos terem que enviar suas informações para um servidor central como o CO-OP, depois do servidor ter criado o conhecimento sobre o tráfego, esse conhecimento é disseminado para os veículos, para que os mesmos possam calcular uma rota alternativa. Sendo assim, EcoTrec consegue reduzir o número de mensagens transmitidas e colisões de pacotes em relação ao



(a) Cobertura

(b) Número de mensagens

(c) Colisões de pacotes

(d) Atraso

Figura 4. Resultados avaliação de escalabilidade.

CO-OP em aproximadamente 30% e 40% respectivamente (veja Figuras 4(b) e 4(c)), consequentemente, EcoTrec consegue um ligeiro aumento na cobertura, atingindo 75% com 20% da densidade total de veículos. Entretanto, assim como o CO-OP, o EcoTrec não implementa nenhum mecanismo para reduzir o número de mensagens transmitidas, sendo assim, o mesmo ainda introduz um alto *overhead* para o sistema, apresentando um atraso médio de aproximadamente 7 minutos com 100% da densidade original (veja Figura 4(d)).

Utilizando uma abordagem distribuída, CARTIM apresenta uma solução escalável, onde o mesmo atinge uma cobertura de aproximadamente 90% para todas as porcentagens de densidade. Além disso, como CARTIM não cria um conhecimento sobre a condição de tráfego, apenas detecta o nível de congestionamento da via em que o veículo está, o mesmo apresenta uma redução significativa na quantidade de mensagens transmitidas, reduzindo em até 95% e 92% quando comparado com CO-OP e EcoTrec, respectivamente. Entretanto, CARTIM não implementa nenhum mecanismo de agregação, sendo assim, muitos veículos na mesma via podem gerar mensagens redundantes sobre o congestionamento da mesma (veja Figuras 4(b) e 4(c)), adicionalmente, CARTIM não verifica o interesse dos veículos nas mensagens disseminadas (classificação do congestionamento), onde muitos veículos podem receber uma mensagem sem ter interesse na mesma. Contudo, como resultado de sua escalabilidade, CARTIM não introduz um *overhead* indesejado para o sistema, apresentando um atraso médio de menos de 1 segundo para todas as porcentagens de densidade (Figura 4(d)).

Por fim, os resultados do GTE mostram sua eficiência e escalabilidade. Gerando o conhecimento apenas das áreas críticas, as quais realmente degradam a eficiência do tráfego, juntamente com a abordagem de segmentação da mesma em áreas menores, assim como os mecanismos de agregação de conhecimento e de compartilhamento oportunista, o qual evita a transmissão de mensagens redundantes e desnecessárias, entregando o conhecimento apenas

para veículos que realmente possuem interesse no mesmo (veículos dentro da AA).

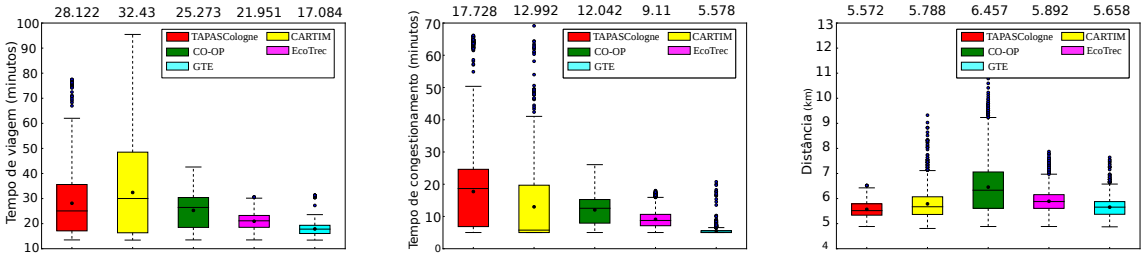
Adicionalmente, GTE reduz drasticamente o número de mensagens transmitidas, apresentando uma redução de mais de 50% quando comparado com CARTIM, 97% em relação ao EcoTrec e 98% em relação ao CO-OP. Consequentemente, GTE apresenta uma cobertura maior que 95% para todas as porcentagens da densidade da densidade original. Além disso, a eficiência do mecanismo de agregação implementado reduz a transmissão de mensagens redundantes, assim reduzindo também o número de colisões de pacotes (veja Figura 4(c)). Sendo assim, o GTE não produz um *overhead* indesejado para o sistema, apresentando um atraso médio de menos de $0,5$ segundos para todas as porcentagens de densidade. Portanto, o GTE consegue construir um conhecimento preciso, transmitindo um menor número de mensagens, maximizando a cobertura e utilização da rede, sem introduzir um *overhead* indesejado.

É importante salientar a baixa escalabilidade apresentada pelas soluções CO-OP e EcoTrec, onde quanto maior a densidade de veículos, maior é o número de mensagens transmitidas e colisões de pacotes, os quais impactam diretamente no desempenho do sistema, assim minimizando a cobertura e introduzindo um *overhead* indesejado.

4.3. Análise de gerência do tráfego

Nesta seção, avaliamos a eficiência da gerência de tráfego do GTE comparando-o com três soluções CARTIM [Araujo et al. 2014], EcoTrec [Doolan and Muntean 2016], CO-OP [de Souza et al. 2015]. É importante salientar que TAPASCologne é o tráfego normal de veículos, sem aplicar nenhum mecanismo para gerenciar o tráfego. Sendo assim, as métricas avaliadas foram: *(i)* Tempo de viagem; *(ii)* Tempo de congestionamento; e *(iii)* Distância percorrida. Para obtermos o congestionamento real do cenário, todas as soluções foram simuladas com 100% da densidade total de veículos (para a região selecionada de 25 km^2).

A Figura 5 apresenta os resultados para todas as métricas avaliadas. Em particular, a Figura 5(a) apresenta os resultados para o tempo de viagem, onde podemos ver que o tráfego original do cenário de TAPASCologne tem um tempo de médio de viagem aproximadamente 28 minutos, onde, em 68% desse tempo os veículos permanecem presos em congestionamentos (veja Figura 5(b)). Como na mobilidade original nenhuma alteração na rota é feita, TAPASCologne apresenta a menor distância percorrida, onde a mesma é de aproximadamente $5,5$ km^2.



(a) Tempo de viagem (b) Tempo de congestionamento (c) Distância percorrida

Figura 5. Resultados análise gerência de tráfego.

CARTIM, apesar de implementar uma solução distribuída, ele aumenta o tempo de viagem em aproximadamente 15% como consequência do seu mecanismo para evitar congestionamentos, o qual não possui conhecimento sobre as reais condições de tráfego. Consequentemente, CARTIM pode criar congestionamentos em outras áreas e também terá que evita-lás, assim, aumentando a distância percorrida (veja Figura 5(c)). Entretanto, CARTIM reduz o tempo de congestionamento em 25% quando comparado com o tráfego real (TAPASCologne).

Em contra partida, CO-OP emprega uma solução centralizada não escalável, portanto, devido ao *overhead* introduzido, o mesmo pode fazer detecções errôneas, como também muitos veículos podem não receber uma rota alternativa, como resultado da sobrecarga da rede. Dessa forma, CO-OP apresenta uma redução de apenas 15% em relação a mobilidade original (TAPASCologne). Entretanto, mesmo com o *overhead* introduzido, CO-OP atinge uma redução de 30% no tempo de congestionamento (veja Figura 5(b)). Utilizando uma abordagem diferente de CARTIM e CO-OP, EcoTrec implementa uma solução híbrida, a qual apresenta uma baixa escalabilidade devido a grande quantidade de mensagens transmitidas entre veículos e servidor central. Porém, diferente do CO-OP, os veículos são responsáveis por calcular suas rotas alternativas com base na condição de tráfego fornecida pelo servidor. Sendo assim, EcoTrec introduz um *overhead* menor que o CO-OP, portanto, o mesmo consegue uma melhor gerência do tráfego, reduzindo em aproximadamente 28% o tempo de viagem e em 45% o tempo de congestionamento, quando comparado com a mobilidade original (TAPASCologne). Para realizar essa melhor gerência do tráfego, tanto CO-OP quanto EcoTrec aumentam a distância percorrida pelos veículos, porém, devido ao sugestão de rota cooperativa implementa pelo CO-OP, o mesmo aumenta a distância percorrida em apenas 10%, enquanto EcoTrec apresenta um aumento de 20% (veja Figura 5(c)).

Por fim, GTE apresenta o menor tempo de viagem e tempo de congestionamento, apresentando uma redução de 36% e 68%, comparado com a mobilidade original, 44% e 57% quando comparado com CARTIM, 30% e 55% quando comparado com CO-OP e, 15% e 40% quando comparado com EcoTrec (veja Figura 5(b) e Figura 5(a)). Tais reduções são resultados da sua alta escalabilidade, a qual não introduz nenhum *overhead* para a solução, e também devido ao conhecimento preciso entregue para os veículos dentro de áreas críticas.

Graças à sua alta escalabilidade, e consequentemente seu baixo *overhead* GTE consegue prover o conhecimento da área crítica para os veículos potencialmente *on-the-fly*, assim, os veículos conseguem detectar congestionamentos e agir antes das outras soluções [de Souza et al. 2015, Doolan and Muntean 2016]. O CARTIM não introduz *overhead* para o sistema, porém o mesmo não tem conhecimento sobre a condição de tráfego e, portanto, os veículos acabam tendo uma má orientação de rota.

O baixo overhead é resultado de sua abordagem totalmente distribuída e dos mecanismos eficientes de agregação e encaminhamento de conhecimento, os quais conseguem prover um conhecimento preciso sobre a condição do tráfego para todos os veículos em uma área crítica, permitindo-os a calcular rotas alternativas para melhorar a eficiência do tráfego. O alto *overhead* introduzido pelas soluções CO-OP e EcoTrec é resultado de ambos utilizarem um servidor central para construir o conhecimento sobre condição de tráfego.

5. Conclusão

Congestionamento se tornou um problema recorrente afetando vários aspectos na sociedade. Porém, muitas soluções para lidar com esse problema apresentam limitações, como a falta de escalabilidade, ou até mesmo a falta de conhecimento sobre o tráfego. Sendo assim, foi proposto GTE, um TMS totalmente distribuído e escalável, baseado em compartilhamento oportunista para lidar com congestionamentos, permitindo que os veículos possuam um conhecimento sobre a condição de tráfego, para que os mesmos possam detectar vias congestionadas e calcularem uma rota alternativa para melhorar a eficiência do tráfego. GTE foi comparado com outras soluções existentes para avaliar sua escalabilidade e eficiência em gerenciar o tráfego.

Os resultados mostraram uma performance superior do GTE em todas as métricas avaliadas, em comparação com as soluções CARTIM, CO-OP e EcoTrec. Como trabalho futuro,

pretende-se analisar como o tamanho da área crítica, área de ancoragem e área de encaminhamento impacta no desempenho da solução. Além disso, pretendemos estender a solução para lidar com congestionamentos causados por eventos inesperados, e também sugerir rotas seguras.

6. Agradecimentos

Os autores gostariam de agradecer o apoio financeiro concedido da FAPESP por meio do processo nº 2015/07538-1, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

Referências

Araujo, G., Queiroz, M., Duarte-Figueiredo, F., Tostes, A., and Loureiro, A. (2014). Um protocolo de identificação e minimização de congestionamentos de trafego para redes veiculares. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - (SBRC)*.

Castro, A. A. V., Serugendo, G. D. M., and Konstantas, D. (2008). Hovering information – self-organizing information that finds its own storage. In *Spring, 2008 Autonomic Communication*, pages 111 – 145.

de Souza, A. M., Guidoni, D., Botega, L. C., and Villas, L. A. (2015). Co-op: Uma solução para a detecção, classificação e minimização de congestionamentos de veículos utilizando roteamento cooperativo. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - (SBRC)*.

de Souza, A. M. and Villas, L. A. (2016). A fully-distributed traffic management system to improve the overall traffic efficiency. In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '16, pages 19–26, New York, NY, USA. ACM.

de Souza, A. M., Yokoyama, R., Boukerche, A., Maia, G., Cerqueira, E., Loureiro, A. A., and Villas, L. A. (2016). Icarus: Improvement of traffic condition through an alerting and re-routing system. *Computer Networks*, 110:118 – 132.

Djahel, S., Doolan, R., Muntean, G.-M., and Murphy, J. (2015). A Communications-Oriented Perspective on Traffic Management Systems for Smart Cities: Challenges and Innovative Approaches. *IEEE Communications Surveys Tutorials*, 17(1):125–151.

Doolan, R. and Muntean, G. M. (2016). Ecotrec: A novel vanet-based approach to reducing vehicle emissions. *IEEE Transactions on Intelligent Transportation Systems*, PP(99):1–13.

Hyytiä, E., Virtamo, J., Lassila, P., Kangasharju, J., and Ott, J. (2011). When does content float? characterizing availability of anchored information in opportunistic content sharing. In *INFOCOM, 2011 Proceedings IEEE*, pages 3137–3145.

Karagiannis, G., Altintas, O., Ekici, E., Heijenk, G., Jarupan, B., Lin, K., and Weil, T. (2011). Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions. *Communications Surveys Tutorials, IEEE*, 13(4):584–616.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., et al. (1983). Optimization by simmulated annealing. *science*, 220(4598):671–680.

Pan, J., Popa, I. S., and Borcea, C. (2016). Divert: A distributed vehicular traffic re-routing system for congestion avoidance. *IEEE Transactions on Mobile Computing*, PP(99):1–1.

Wang, M., Shan, H., Lu, R., Zhang, R., Shen, X., and Bai, F. (2015). Real-time path planning based on hybrid-vanet-enhanced transportation system. *Vehicular Technology, IEEE Transactions on*.

Planejando a Infraestrutura de Comunicação para a Distribuição de Mídias em Tempo Real para Veículos

Leonardo A. A. Pereira[1], **João F. M. Sarubbi**[2],
Cristiano G. Pitangui[1], **Cristiano M. Silva**[1]

[1]Universidade Federal de São João Del-Rei – Minas Gerais – Brasil

[2]Centro Federal de Educação Tecnológica – Minas Gerais – Brasil

leonardoaapereira@gmail.com, joao@decom.cefetmg.br

{pitangui.cristiano, cristiano}@ufsj.edu.br

Abstract. *In this work we propose Sigma-Deployment as solution for communication in vehicular networks that relies on V2I communication. The solution uses the "inter-contact"time between vehicles and infrastructure and the speeds of consumption and transfer of data to alocate communication units over the road mesh. These parameters ensure that a certain fraction of vehicles maintain frequent contact with the network and that these vehicles always have data to consume. The Sigma-Deployment is compared with the Densest Locations strategy and it shows improvements.*

Resumo. *Este trabalho propõe a Deposição-Sigma como solução para comunicação em redes veiculares que utilizem de comunicação V2I. A solução utiliza o tempo entre contatos de veículos com a infraestrutura e as velocidades de transferência e consumo de dados para alocar RSUs ao longo da malha rodoviária. Estes parâmetros asseguram que certa fração de veículos mantêm contato frequente com a rede e que estes veículos possuam sempre dados para consumir. A Deposição-Sigma é comparada com a estratégia Densest Locations e os resultados demonstram melhorias.*

1. Introdução

Nos Sistemas Inteligentes de Transporte (SIT), veículos, pedestres, ciclistas, usuários do transporte públicos, semáforos, e sensores de via são integrados em uma única rede, e tais elementos atuam como sensores da mobilidade urbana coletando grandes quantidades de dados que são processados e devolvidos aos motoristas na forma de recomendações de tráfego [Silva et al. 2015a]. Nestes sistemas, a comunicação é questão central, entretanto, a grande mobilidade dos veículos impõe grandes desafios.

Estratégias semelhantes as utilizadas em redes celulares não demonstram ser adequadas ao contexto de redes veiculares, pois demandam cobertura ao longo de toda a viagem, enquanto veículos podem atravessar áreas de cobertura localizadas ao longo da malha viária. Semelhantemente, estratégias baseadas na latência e largura de banda tornam-se dependentes da localização e oscilam de acordo com a posição do nó (veículo) a ser avaliado. Através da disponibilização de uma infraestrutura de comunicação, aqui chamada de RSU[1], pode-se criar uma rede veicular que permite a coleta e disseminação de dados de tráfego.

[1]Refere-se ao termo "Unidade de beira de estrada", do inglês "Roadside Unit"

Neste trabalho é apresentada uma solução para alocação de RSUs em redes veiculares que utilizem comunicação V2I[2], e que necessitem de regularidade no tempo entre contatos (tempo que um veículo passa sem se conectar à uma RSU) dos veículos com a infraestrutura. Esta estratégia também garante que para a fração de veículos que manter a regularidade citada, haverá dados disponíveis para consumo durante toda suas trajetórias.

Este trabalho propõe a métrica *Deposição-Sigma* (σ) para caracterizar o desempenho de uma dada distruibuição de RSUs. Uma dada distribuição $\sigma[\rho|\tau|v_c|v_d]$ garante que ρ porcento dos veículos passam por RSUs em intervalos de tempo não maiores que τ segundos, e que estes veículos mantêm sua carga de dados acima de zero, sendo esta controlada pelas velocidades de download v_d e consumo v_c.

O presente trabalho apresenta a métrica *Deposição-Sigma* em conjunto com a estratégia de deposição *Sigma-*ϕ para realizar a deposição das RSUs baseada no tempo entre contatos dos veículos com a infraestrutura e na oscilação da quantidade de dados obtido da rede. *Sigma-*ϕ demonstra-se eficiente em oferecer regularidade de contatos e transferência de informações dentro de uma rede veicular, garantindo o tempo entre contatos para determinada fração de veículos de forma que esses veículos consigam receber dados ao longo de todo o seu trajeto. *Sigma-*ϕ oferece ao administrador da rede a capacidade de monitorará-la com alto nível de assertividade e, por meio da propagação da informação, permite a detecção de incidentes e pronta resposta em recomendações de tráfego.

Compara-se a estratégia proposta com a deposição intuitiva de implantar as RSUs nos locais de maior concentração de veículos (DL ou *Densest Locations*). Os resultados experimentais demonstram que *Sigma-*ϕ reduz o número de RSUs necessárias para obter a mesma cobertura proporcionada pela *Densest Locations (DL)* (15,49% a 41,58%), sem gerar grande impacto no desempenho da rede, ou mesmo zero em alguns casos.

Este trabalho diferencia-se de outros por considerar a influência das velocidades de download e consumo de dados na alocação das RSUs, garantindo que uma determinada fração de veículos mantenha sua carga de dados acima de zero durante todo a trajetória, e por garantir a regularidade de contatos e transferência dentro da rede veicular em questão.

O restante deste trabalho encontra-se estruturado da seguinte forma: A Seção 2 apresenta os trabalhos relacionados. A seção 3 introduz a *Deposição-Sigma*. A seção 4 introduz o algorítmo utilizado na estratégia *Sigma-*ϕ. A seção 5 apresenta a estratégia *DL* usada na comparação de resultados. A seção 6 apresenta os resultados obtidos através de experimentos. A seção 7 conclui o trabalho.

2. Trabalhos Relacionados

A literatura apresenta estudos analíticos sobre garantias mínimas de qualidade da rede veicular. Tipicamente, esses trabalhos apresentam limites máximos para a transmissão de dados. Como exemplo, [Zheng et al. 2010] apresenta a avaliação de uma estratégia de implantação de infraestrutura através do conceito de oportunidade de contato. Já o trabalho [Lee and Kim 2010] propõe uma heurística para a implantação de infraestrutura que busca ampliar a conectividade dos veículos e reduzir o intervalo sem conexões. Por sua vez, [Nekoui et al. 2008] propõe a implantação de infraestrutura com base na definição de capacidade de transporte.

[2]Comunicação Veículo-Infraestrutura

Em termos de estratégias geométricas, [Cheng et al. 2013] propõem uma heurística para solucionar o problema da máxima cobertura, enquanto que [Patil and Gokhale 2013] empregam diagramas de Voronoi. Já em termos de modelos de otimização, [Aslam et al. 2012] usam Programação Inteira Binária para o problema de alocação de infraestrutura. A partir da eliminação de vias secundárias, as vias principais são modeladas como uma grade (*grid*). [Wu et al. 2012] consideram um cenário de autoestrada com múltiplas pistas, e abordam também a comunicação de múltiplos saltos.

[Trullols et al. 2010] e [Cataldi and Harri 2011] propõem a modelagem de alocação de RSUs como um problema de máxima cobertura, assumindo-se a premissa de conhecimento prévio das trajetórias dos veículos. Já Silva et al. [Silva et al. 2015b] propõem a modelagem via problema da máxima cobertura probabilística (PMCP).

Diferentemente das abordagens anteriores, a estratégia proposta neste trabalho considera o tempo entre contatos e o controle da carga de dados do veículos para identificar os melhores locais para implantação das RSUs, permitindo o amplo sensoriamento do tráfego, bem como a elaboração de estratégias sofisticadas para entrega de conteúdos em redes veiculares.

3. Apresentação da Deposição-Sigma

A *Deposição-Sigma* (σ) é uma métrica utilizada para caracterizar o desempenho de uma distribuição de RSUs ao longo de uma malha rodoviária. Estas RSUs fazem parte da rede veicular local e utilizam protocolo de comunicação V2I.

A alocação de RSUs é modelada de acordo com o tempo entre contatos e as velocidades de download e consumo de dados. Esta deposição tem como objetivo oferecer regularidade de contatos e a disseminação de informações para uma determinada fração de veículos de modo que, através desta disseminação é possível a detecção de incidentes de trânsito e a pronta resposta em recomendações de tráfego. Deste modo, a *Deposição-Sigma* faz com o que o administrador da rede torne-se capaz de monitorar o fluxo do tráfego com alto nível de assertividade.

O tempo entre contatos τ regula a velocidade com que os dados são disseminados. Uma redução no valor de τ acarreta na redução de RSUs utilizadas, porém diminui a precisão de controle da rede, visto que as informações irão levar mais tempo para atingir os usuários. As velocidades de download v_d (velocidade cujo os dados são baixados) e consumo v_c (velocidade no qual os dados se esgotam, por exemplo, o tempo necessário para executar um vídeo baixado) de dados controlam a carga[3] total do veículo Q. Se um veículo possuir carga positiva significa que, mesmo fora da cobertura de uma RSU, ele possui informações sobre a rede recebidas de RSUs anteriores.

A diferença entre a velocidade de download e a de consumo é chamada de velocidade de recarga. Para que se tenha uma configuração válida, é necessário que a velocidade de download seja sempre maior que a de consumo, caso contrário a velocidade de recarga será menor ou igual a zero. Deste modo, quando um veículo está dentro da área de cobertura de uma RSU, este está ganhando mais carga do que consome pois $v_d > v_c$, ou seja, está recarregando. Quando está fora da área de cobertura, o mesmo somente perde carga. Em redes com valores de download e consumo próximos mais RSUs são

[3]Quantidade de dados disponível.

necessárias, pois a velocidade de recarga será lenta. Parte da caracterização proposta pela métrica *Deposição-Sigma* é expressa pelo gráfico Sigma a seguir. O eixo Y expressa o desempenho máximo[4] da rede para os valores de τ correspondentes no eixo x.

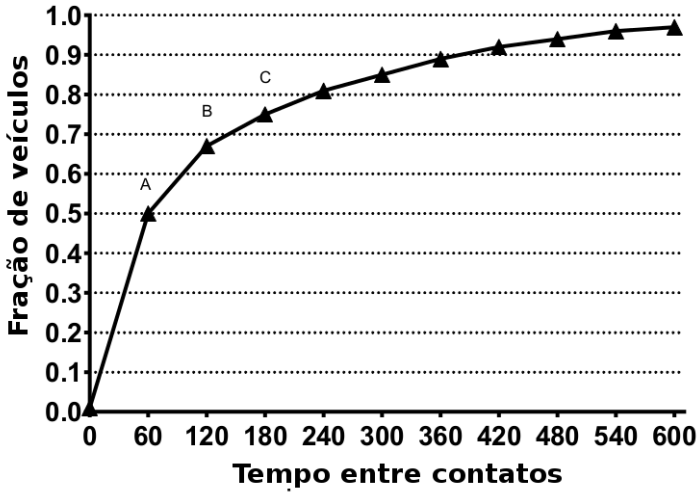
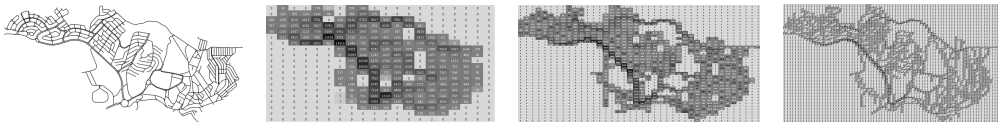


Figura 1. Relação entre tempo entre contatos e a porcentagem de veículos cobertos.

A figura 1 apresenta a análise de desempenho para a deposição $\sigma[50|60|1.00|2.00]$. O ponto A indica que 50% dos veículos realizam contatos em intervalos menores ou iguais a 60 segundos. Já o ponto B indica que 68% dos veículos realizam contatos em intervalos menores ou iguais a 120 segundos. Por fim, o ponto C indica que 76% dos veículos realizam contatos em intervalos menores ou iguais a 180 segundos.



(a) Malha Rodoviária (b) Partição 20 x 20 (c) Partição 40 x 40 (d) Partição 20 x 20

Figura 2. Exemplos de particionamentos da malha rodoviária

A fim de generalizar as diferentes formas e topografias das malhas rodoviárias, propõe-se o particionamento da área em um conjunto de ψ x ψ células de mesmo tamanho. Caso o projeto necessite de maior/menor precisão, aumenta-se/diminui-se o valor de ψ, o que diminui/aumenta o tamanho de cada célula. Ao se realizar este particionamento, não se objetiva representar raios de cobertura, mas sim dividir a malha viária em células de mesmo tamanho, assume-se então que uma RSU é capaz de cobrir a área de uma célula urbana independente de fatores externos tais como relevo, construções bloqueando sinais, etc. Por fim, a partição diminui a quantidade de locais a serem avaliados, e consequentemente reduz o esforço computacional. As Figuras 2(a) - 2(d) representam exemplos de possíveis particionamentos.

[4]Porcentagem máxima de veículos que pode ser coberta de acordo com os parâmetros iniciais.

4. Heurística Sigma

Nesta seção apresenta-se a estratégia *Sigma-ϕ* utilizada pela *Deposição Sigma* para alocar as RSUs. Em *Sigma-ϕ*, um veículo *v* é considerado coberto quando:

- O tempo entre contatos de *v* não ultrapassou o valor τ;
- A carga *Q* de *v* manteve-se acima de zero durante toda a trajetória do veículo.

A heurística recebe como parâmetros:

- A malha rodoviária (M);
- A porcentagem de veículos à ser coberta (ρ);
- O tempo limite entre contatos (τ);
- A velocidade de consumo de dados (v_c);
- A velocidade de download de dados (v_d);
- O fluxo de veículos (V).

Sigma-ϕ é composta por três etapas, listadas abaixo e detalhadas logo em seguida.

1 - **Deposição inicial:** etapa responsável por alocar as unidades de comunicações em suas posições iniciais, utilizando os pontos com maior tempo de permanência de veículos, até satisfazer os parâmetros iniciais;
2 - **Busca local:** as RSUs alocadas em 1 são movidas em sua vizinhança buscando-se maximizar o total de veículos cobertos. Ao final, o número de RSUs não é alterado, no entanto, a quantidade de veículos cobertos pode ser maior que o valor inicial de ρ estabelecido;
3 - **Remoção de unidades extras:** tendo que o passo 2 obteve sucesso em aumentar a quantidade de veículos cobertos, este passo remove as RSUs em excesso, selecionando as que geram menor impacto no total de veículos cobertos, até que se atinja o objetivo de ρ por cento dos veículos.

4.1. Deposição Inicial

Esta etapa realiza a deposição inicial de antenas, gerando o conjunto solução inicial Υ. Este conjunto, mesmo sendo uma solução válida para o problema, ainda não estará otimizado e ainda será processado nas etapas Busca Local e Remoção de Unidades Extras.

O algoritmo 1 demonstra esta etapa. Recebe-se como entrada, a malha rodoviária já particionada *(M)*, o fluxo de veículos *(V)*, o tempo limite entre contatos (τ), a fração de veículos a ser coberta (ρ) e as velocidades de download (v_d) e de consumo (v_c). Ao final, é obtido o conjunto Υ indicando as RSUs alocadas e suas posições.

A estratégia Sigma-ϕ começa selecionando o primeiro veículo V_i. Supõe-se que este veículo tem carga inicial igual a $v_c \times \tau$ e seu tempo inicial entre contatos igual a zero, deste modo o veículo é capaz de percorrer τ segundos sem a necessidade de encontrar uma RSU. A cada célula que este veículo percorre, o mesmo permanece *t* segundos na mesma, caso esta célula não possua uma RSU, subtrai-se de sua carga $v_c \times t$, adiciona-se ao tempo entre contatos t, e esta célula é guardada em um conjunto temporário ζ. Caso esta célula possua uma RSU, adiciona-se à carga do veículo $(v_d - v_c) \times t$, o tempo entre contatos é zerado, e o conjunto ζ é limpo.

Caso o tempo entre contatos ultrapasse o valor de τ, ou a carga atinja valores menores ou iguais a zero, todas as células em ζ passam a ser consideradas candidatas

Algoritmo 1 Deposição Inicial

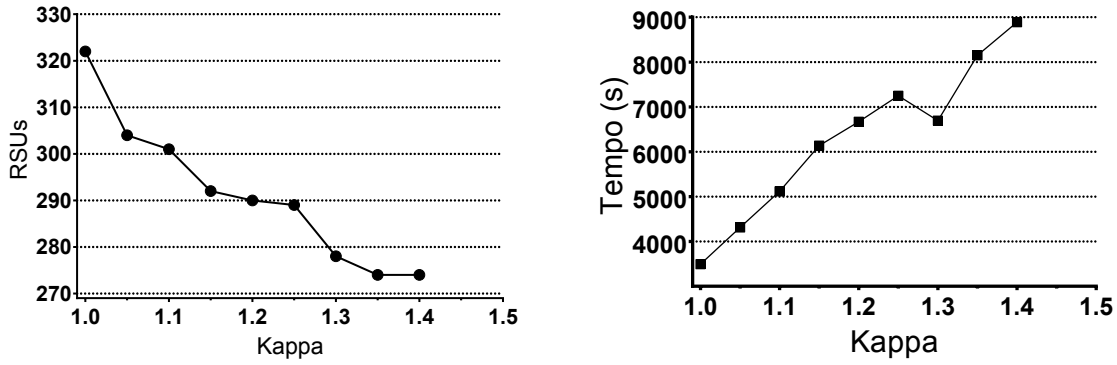
Entrada: $M, V, \tau, \rho, v_c, v_d$;
Saída: Υ
1: $\Upsilon \leftarrow \emptyset$; ▷ O conjunto soluções é iniciado vazio.
2: VeículosCobertos $\leftarrow 0$; ▷ Inicia a variável quantidade de veículos cobertos.
3: **repeat**
4: **Para Todo(a)** Veículo dentro do conjunto V **Faça**
5: $T \leftarrow 0$; ▷ Zere o tempo sem RSU.
6: $Q \leftarrow v_c \times \tau$; ▷ Defina a carga inicial.
7: $\zeta \leftarrow \emptyset$; ▷ Esvazie o vetor temporário de RSUs.
8: **Para Todo(a)** Célula C trafegada pelo veículo **Faça**
9: **Se** Célula possui RSU **Então**
10: $Q \leftarrow Q + (v_d - v_c) \times \tau$; ▷ Atualize a carga do veículo.
11: $T \leftarrow 0$; ▷ Zere o tempo sem RSU.
12: $\zeta \leftarrow \emptyset$; ▷ Esvazie o vetor temporário de RSUs.
13: **Senão**
14: $Q \leftarrow Q - v_c \times \tau$; ▷ Remova carga do veículo.
15: $T \leftarrow T$ + TempoNaCelula(); ▷ Incremente o tempo sem RSU.
16: $\zeta \leftarrow C$; ▷ Guarde a célula no vetor temporário de RSUs.
17: **Fim Se**
18: **Se** $T > \tau$ **ou** $Q \leq 0$ **Então**
19: **Para Todo(a)** Célula no vetor temporário de RSUs **Faça**
20: Incremente a pontuação da célula de acordo com o tempo;
21: **Fim Para**
22: $\zeta \leftarrow \emptyset$; ▷ Esvazie o vetor temporário de RSUs.
23: **Fim Se**
24: **Fim Para**
25: **Fim Para**
26: Selecione a célula com maior pontuação;
27: BuscaLocal(); ▷ Realize a busca local nesta célula
28: $\Upsilon \leftarrow$ MelhorCélula; ▷ A melhor posição é adicionada ao conjunto solução.
29: LimpaPontuações(); ▷ Limpa a pontuação de todas as células.
30: VeículosCobertos $\leftarrow$ ComputaVeículosCobertos(M, v_c, v_d, ρ, τ); ▷ Atualiza a quantidade de veículos cobertos.
31: **until** (VeículosCobertos $< \rho$); ▷ Enquanto o parâmetro ρ não for atingido.
32: $\Phi \leftarrow$ ComputaTotalRSUs(Υ) $\times\kappa$;
33: **Enquanto** (ComputaTotalRSUs(Υ) $< \Phi$) **Faça** ▷ Enquanto o total de RSUs Φ não for atingido.
34: Repita de 4 a 29; ▷ Repete o loop anterior, sem a necessidade de calcular os veículos cobertos.
35: **Fim Enquanto**
36: **Retorne** Υ

a receberem RSUs. Para isto, todas tem sua pontuação incrementada de acordo com o tempo que o veículo passou naquela célula. Em seguida, o vetor ζ é novamente zerado.

Após repetir o procedimento para todos os veículos, a célula com maior pontuação é selecionada e, todas as posições as seu redor (com distância máxima de 1 quadrante) são avaliadas de modo que a posição que obter o maior número de veículos cobertos quando adicionado uma RSU na mesma é adicionada ao conjunto solução Υ. Em seguida, a pontuação de todas as células é zerada. Este procedimento é realizado até que o total de veículos cobertos atinja o valor de ρ.

Em seguida, são alocadas κ porcento antenas a mais, com objetivo de aumentar o tamanho do conjunto solução Υ. Esta expansão acarreta na otimização da etapa 2 (Busca Local) e, consequentemente, apresenta melhores resultados finais. Entretanto, quanto maior o valor de κ maior o tempo necessário para a conclusão do algoritmo.

A figura 3(a) demonstra a relação entre κ e o número final de RSUs, e a figura 3(b) entre κ e o tempo gasto em segundos. Para os valores de κ = 1.35 e 1.4 o número final de RSUs permaneceu o mesmo, entretanto o tempo aumentou de aproximadamente 750 segundos. Considera-se então 1.35 como o valor limite, ou seja, acima deste ponto, a diferença no número de RSUs será desprezível comparada a diferença no tempo gasto.



(a) (b)

Figura 3. Comparação dos valores de κ em relação ao número de RSUs e ao tempo gasto.

4.2. Busca Local

Após a etapa 1, tem-se o conjunto solução Υ, este conjunto é composto pelas unidades de comunicações e suas posições na malha rodoviária. Esta etapa tem por objetivo otimizar as posições de todas as RSUs pertencentes a Υ. Para isso é utilizada a estratégia conhecida como subida de encosta (*Hill Climbing*).

Algoritmo 2 Busca Local

Entrada: $\Upsilon, M, V, \tau, \rho, v_c, v_d$;
Saída: Υ
1: **repeat**
2: HouveMudança = 0;
3: **Para Todo(a)** RSU$_i$ no conjunto Υ **Faça**
4: cobertualOriginal = ComputaVeículosCobertos($\Upsilon, M, v_c, v_d, \rho, \tau$);
5: PosiçãoOriginal = RSU$_i$.pos;
6: **Para Todo(a)** (Posições na Vizinhança) **Faça**
7: Mova a RSU$_i$ para a nova posição;
8: TVL[NovaPosição] = ComputaVeículosCobertos($\Upsilon, M, v_c, v_d, \rho, \tau$);
9: **Fim Para**
10: MelhorResultado = TVL.MaiorValor();
11: MelhorPosição = TVL.PosiçãoDeMaiorValor();
12: **Se** (MelhorResultado $>$ coberturaOriginal) **Então**
13: RSU*i*.pos = MelhorPosição;
14: HouveMudanca = 1;
15: **Senão**
16: RSU*i*.pos = PosiçãoOriginal;
17: **Fim Se**
18: **Fim Para**
19: **until** (HouveMudança == 1);

O algoritmo seleciona a primeira antena (RSU*i*) do conjunto Υ e a move em sua vizinhança. Em cada nova posição o total de veículos cobertos é recalculado e seu resultado guardado no vetor temporário *TVL*. Ao se mover para todas as posições possíveis em sua vizinhança, o maior valor de *TVL* é comparado com a quantidade inicial de veículos cobertos, caso seja maior, a RSU é movida para a posição que gerou este novo valor, caso seja menor ou igual, a RSU permanece na sua posição original. Este processo é repetido com todas as antenas do conjunto Υ.

Entretanto, como o algoritmo verifica as RSUs de maneira ordenada, quando se modifica a posição da RSU RSU*i*, para todas as unidades anteriores a *i* podem surgir novas

posições com melhores resultados. Logo, a etapa de Busca Local é realizada enquanto houver trocas dentro do conjunto Υ.

O algoritmo 2 formaliza esta etapa. Recebe-se como entrada, a malha rodoviária já particionada *(M)*, o fluxo de veículos *(V)*, o tempo limite entre contatos *(τ)*, a fração de veículos a ser coberta *(ρ)* e as velocidades de download (v_d) e de consumo (v_c). Ao final, é obtido o conjunto Υ otimizado.

4.3. Remoção de Unidades Extras

A otimização gerada pela etapa 2 acarreta no aumento da quantidade de veículos cobertos da etapa 1. A etapa 3 tem como proposta a remoção de RSUs em excesso, diminuindo o total de veículos cobertos, porém sem atingir valores abaixo de ρ. Basicamente, para cada RSU, é calculado o impacto gerado no total de veículos cobertos ao se remover a unidade e, então, remove-se a unidade que gera o menor impacto. Este processo é repetido enquanto o total de veículos cobertos for maior que ρ.

Algoritmo 3 Remoção de Unidades Extras

Entrada: $\Upsilon, M, V, \tau, \rho, v_c, v_d$;
Saída: Υ
Enquanto VeículosCobertos ρ **Faça** VeículosCobertos = ComputaVeículosCobertos($\Upsilon, M, v_c, v_d, \rho, \tau$);
  Para Todo(a) (RSU$i \in \Upsilon$) **Faça**
    Υ' $\leftarrow$ Υ - RSUi;
    Impacto[i] = VeículosCobertos - ComputaVeículosCobertos(Υ', M, v_c, v_d, ρ, τ);
  Fim Para VeiculosCobertos = Impacto.MaiorResultado();
  Se VeículosCobertos $> \rho$ **Então**
    $\Upsilon \leftarrow \Upsilon$';
  Senão
    Fim();
  Fim Se
Fim Enquanto
Retorne Υ;

O algoritmo 3 formaliza esta etapa. Recebe-se como entrada, a malha rodoviária já particionada *(M)*, o fluxo de veículos *(V)*, o tempo limite entre contatos *(τ)*, a fração de veículos a ser coberta *(ρ)* e as velocidades de download (v_d) e de consumo (v_c). Ao final, é obtido o conjunto Υ final.

5. Estratégia DL

Com o objetivo de comparar o desempenho da estratégia Sigma-ϕ, utiliza-se a estratégia DL, que consiste em alocar as RSUs nos pontos de maior densidade de tráfego e que ainda não possuem RSU. O algoritmo a seguir formaliza a estratégia DL, que recebe como parâmetros a malha rodoviária (M), o conjunto de veículos (V), o tempo máximo "entre-contatos"(τ) e as velocidades de download (v_d) e de consumo (v_c).

Algoritmo 4 Estratégia DL

Entrada: $M, V, \tau, \rho, v_c, v_d$;
Saída: Υ
Enquanto (VeículosCobertos $< \rho$) **Faça**
  Adicione uma RSU na célula de maior densidade de tráfego;
Fim Enquanto
Retorne Υ;

6. Resultados Experimentais

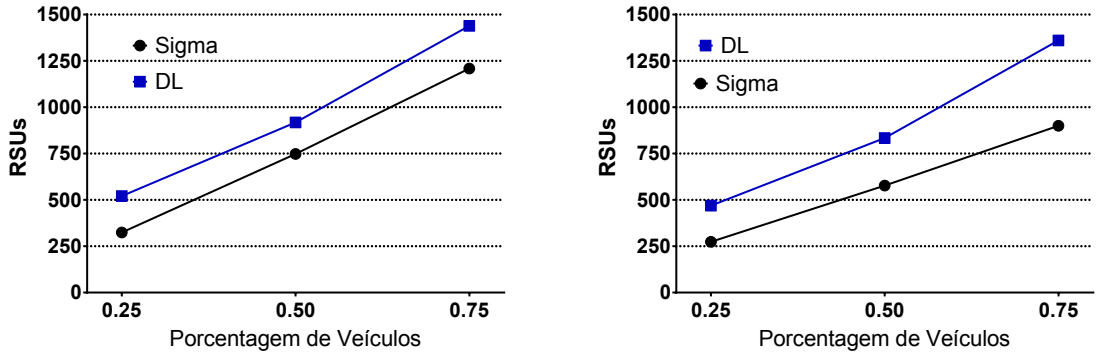
Esta seção apresenta a comparação dos resultados obtidos em *Sigma* e DL. Como fluxo de veículos é utilizado o rastro (*trace*) realístico de mobilidade da cidade de Colônia (Alemanha). O fluxo conta com 7.200 segundos de tráfego e 75.515 veículos. Particionou-se a cidade de Colônia em uma grade de 100x100 células, a fim de se obter células que podem ser cobertas por apenas uma RSU. Através desta partição obteve-se células de 270m x 260m, um raio típico de cobertura para a comunicação veicular, conforme [Teixeira et al. 2014]. Os experimentos são realizados utilizando o simulador SUMO[5].

Para todos os casos de teste foi utilizado $\kappa = 1.35$. Além deste ponto a melhora é insignificante comparado ao tempo de execução do algoritmo. Para comparar o desempenho das duas estratégias, dois resultados requerem maior atenção: a) investimentos necessários para se montar a rede; b) cobertura e QoS oferecida pela rede.

Os **investimentos** podem ser estimados através do número de RSUs utilizadas pela rede. Um alto número de RSUs pode indicar uma rede inviável financeiramente. Já a **cobertura** pode ser analisada através da própria Deposição Sigma que apresenta os níveis de desempenho da rede para diversos cenários.

6.1. Análise de Investimentos

A análise de investimentos compara o número de RSUs utilizadas por cada estratégia.



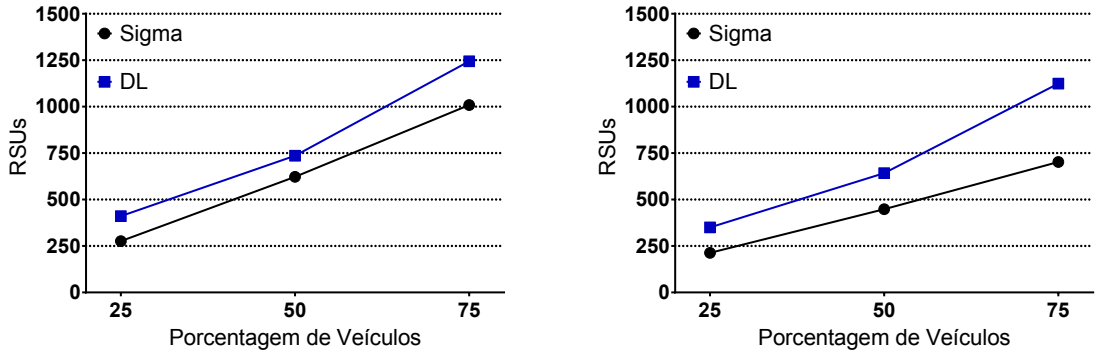
(a) $v_d = 1.25$ (b) $v_d = 2.00$

Figura 4. Caso 1 - Comparação do número de RSUs utilizado em Sigma-ϕ e DL. O eixo x representa a fração de veículos ρ e o eixo y a quantidade de RSUs utilizada.

As figuras 4(a) e 4(b) representam a análise de investimentos entre Sigma-ϕ e DL para o caso em que τ é 60 segundos, a fração de veículos ρ assume os valores de 25%, 50% e 75%, a velocidade de download (v_d) assume 1.25 e 2.00, e a velocidade de consumo (v_c) é fixa em 1. No melhor caso, $\rho = 25\%$, $v_d = 2.00$, *Sigma*-ϕ gasta 274 RSUs, enquanto DL requer 469, ou seja, uma redução superior à 41% no total de RSUs.

As figuras 5(a) e 5(b) representam a análise de investimentos entre Sigma-ϕ e DL para o caso em que τ é 90 segundos, a fração de veículos ρ assume os valores de 25%, 50% e 75%, a velocidade de download (v_d) assume 1.25 e 2.00, e a velocidade de

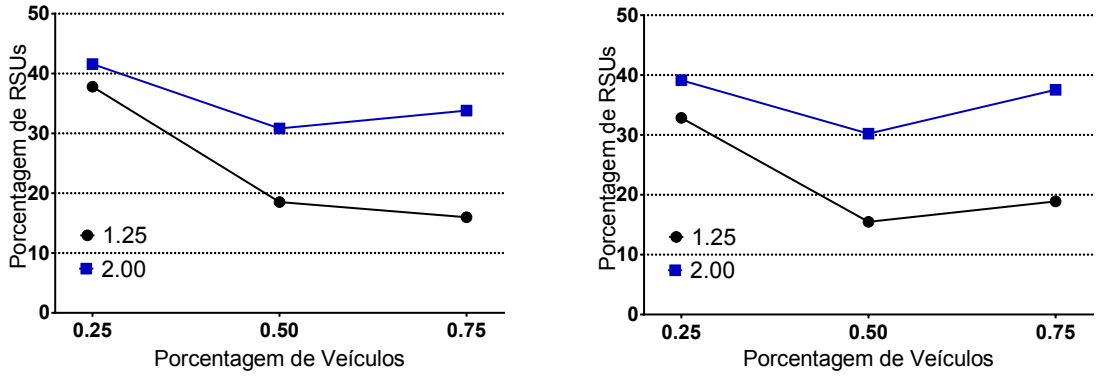
[5]Simulator Sumo: `http://sumo-sim.org`.



(a) $v_d = 1.25$ (b) $v_d = 2.00$

Figura 5. Caso 2 - Comparação do número de RSUs utilizado em Sigma-ϕ e DL. O eixo x representa a fração de veículos ρ e o eixo y a quantidade de RSUs utilizada.

consumo (v_c) é fixa em um. No melhor caso, $\rho = 25\%$, $v_d = 2.00$, Sigma-ϕ gasta 213 RSUs, enquanto DL requer 350, ou seja, uma redução de 39% no total de RSUs.



(a) $\tau = 60$ (b) $\tau = 90$

Figura 6. Porcentagem de RSUs que Sigma-ϕ gasta a menos que DL.

As figuras 6(a) e 6(b) resumem os resultados dos casos anteriores, indicando a porcentagem de RSUs que Sigma-ϕ gasta a menos que DL. Nos casos avaliados, Sigma-ϕ requer de -15% a -41% RSUs que DL. Nos casos citados acima, obseva-se que Sigma foi capaz de obter a mesma cobertura que DL para os parâmetros desejados, porém reduzindo a quantidade de RSUs necessárias, o que acarreta na redução dos inventimentos.

6.2. Cobertura

Parte da análise do desempenho da rede consiste em analisar a qualidade da cobertura oferecida pela mesma. A avaliação desta cobertura é apresentada nos gráficos Sigma-ϕ/DL a seguir. Estes apresentam a distribuição de veículos de acordo com o tempo entre contatos para uma determinada deposição Sigma-ϕ ou DL. O eixo x representa τ enquanto y indica o correspondente valor de ρ. Observa-se que pelo fato de DL utilizar mais RSUs que Sigma-ϕ, ela consegue cobrir mais veículos *Entre 15% e 0%* para valores de τ diferentes daquele passado como parâmetro. Entretanto, no último caso, a diferença cai menos de 5% quando o tempo entre contatos é maior que o τ desejado.

A figura 7(a) se refere às deposições *Sigma* e DL com parâmetros $\tau = 60$, $v_c = 1$, $v_d = 1.25$ e $\rho = 50$. O ponto (x=60, y=5) é comum às duas curvas, pois é o ponto central das deposições. A figura 7(b) refere-se às deposições *Sigma* e DL com parâmetros $\tau = 60$, $v_c = 1$, $v_d = 2.00$ e $\rho = 75$. Novamente, o ponto (x=60, y=75) é comum a ambas as curvas.

As figuras 7(c) e 7(d) realçam as variações em termos da fração de veículos entre *Sigma* e DL para as deposições propostas nas figuras 7(a) e 7(b) respectivamente. Nota-se no segundo caso que esta variação para valores de τ acima de 60 é praticamente nula.

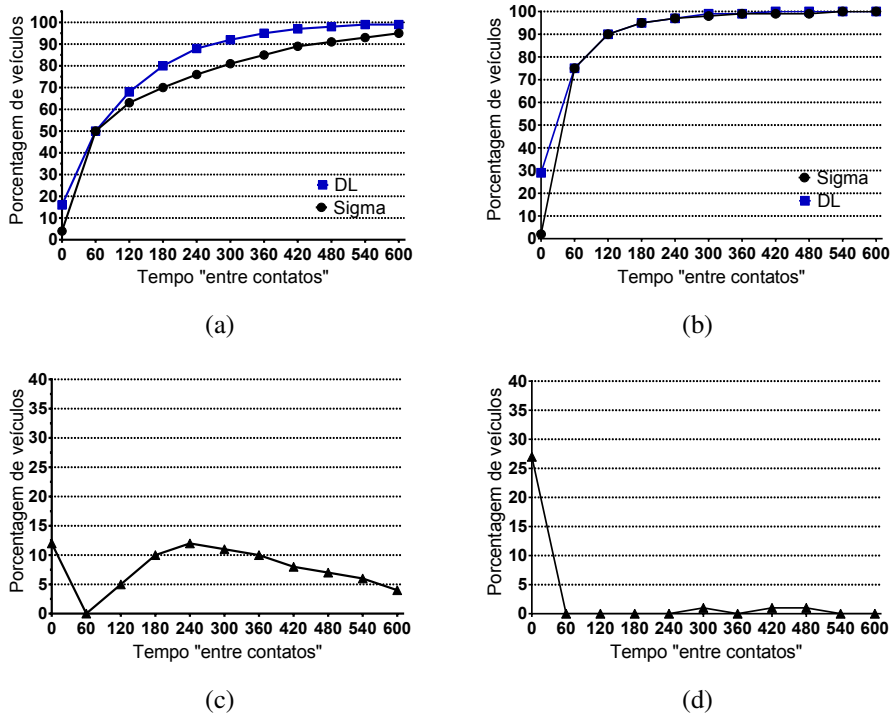


Figura 7. Comparação da qualidade de rede entre Sigma-ϕ e DL. As figuras (a) e (c) representam o desempenho das duas redes. As figuras (b) e (d) representam a diferença de desempenho *(DL - Sigma)* entre redes.

7. Conclusão

O presente trabalho apresentou a métrica *Deposição-Sigma* em conjunto com a estratégia de deposição *Sigma-ϕ* para realizar a deposição das RSUs baseada no tempo entre contatos dos veículos com a infraestrutura e na oscilação da quantidade de dados obtido da rede. Levando-se em consideração os resultados apresentados na seção anterior concluiu-se que *Sigma-ϕ* é mais eficiente que a estratégia *Densest Locations*, visto que *Sigma-ϕ* utiliza uma quantidade menor ou igual de recursos financeiros, com pequeno impacto na qualidade da rede.

Agradecimentos

Este trabalho foi parcialmente financiado por recursos do CNPq, CAPES e FAPEMIG.

Referências

Aslam, B., Amjad, F., and Zou, C. (2012). Optimal roadside units placement in urban areas for vehicular networks. In *Computers and Communications (ISCC), 2012 IEEE Symposium on*, pages 000423–000429. IEEE.

Cataldi, P. and Harri, J. (2011). User/operator utility-based infrastructure deployment strategies for vehicular networks. In *Vehicular Technology Conference (VTC Fall), 2011 IEEE*, pages 1–5.

Cheng, H., Fei, X., Boukerche, A., Mammeri, A., and Almulla, M. (2013). A geometry-based coverage strategy over urban vanets. In *Proceedings of the 10th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, PE-WASUN '13, pages 121–128, New York, NY, USA. ACM.

Lee, J. and Kim, C. (2010). A roadside unit placement scheme for vehicular telematics networks. In Kim, T.-h. and Adeli, H., editors, *Advances in Computer Science and Information Technology*, volume 6059 of *Lecture Notes in Computer Science*, pages 196–202. Springer Berlin Heidelberg.

Nekoui, M., Eslami, A., and Pishro-Nik, H. (2008). The capacity of vehicular ad hoc networks with infrastructure. In *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops, 2008. WiOPT 2008. 6th International Symposium on*, pages 267–272.

Patil, P. and Gokhale, A. (2013). Voronoi-based placement of road-side units to improve dynamic resource management in vehicular ad hoc networks. In *Collaboration Technologies and Systems (CTS), 2013 International Conference on*, pages 389–396.

Silva, C. M., Andre L. L. Aquino, and Wagner Meira Jr (2015a). Deployment of roadside units based on partial mobility information. *Computer Communications*, 60(0):28 – 39.

Silva, C. M., Wagner Meira Jr, and Joao F. M. Sarubbi (2015b). Non-Intrusive Planning of Roadside Infrastructure for Vehicular Networks Without Tracking Individual Vehicles. *IEEE Transactions on Intelligent Transportation Systems*, -(-):–.

Teixeira, F., Silva, V., Leoni, J., Macedo, D., and Nogueira, J. M. S. (2014). "vehicular networks using the ieee 802.11p standard: An experimental analysis". *Vehicular Communications*, 1(2):91 – 96.

Trullols, O., Fiore, M., Casetti, C., Chiasserini, C., and Ordinas, J. B. (2010). Planning roadside infrastructure for information dissemination in intelligent transportation systems. *Computer Communications*, 33(4):432 – 442.

Wu, T.-J., Liao, W., and Chang, C.-J. (2012). A cost-effective strategy for road-side unit placement in vehicular networks. *Communications, IEEE Transactions on*, 60(8):2295–2303.

Zheng, Z., Lu, Z., Sinha, P., and Kumar, S. (2010). Maximizing the contact opportunity for vehicular internet access. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9.

Trilha Principal do SBRC 2017
Sessão Técnica 5
Tolerância a Falhas I

Algoritmo de Difusão Atômica Rápido a Despeito de Colisões Tolerante a Falhas Bizantinas

Rodrigo Q. Saramago[1], Eduardo A. P. Alchieri[2], Tuanir F. Rezende[1], Lasaro Camargos[1]

[1] Faculdade de Ciência da Computação – Universidade Federal de Uberlândia (UFU)

[2]Departamento de Ciência da Computação – Universidade de Brasília (UNB) *

Abstract. *The inefficiency of Consensus-based Atomic Broadcast protocols in the presence of collisions (concurrent proposals) harms their adoption in the implementation of State Machine Replication. Proposals that are not decided in some instance of Consensus (commands not delivered) must be reproposed in a new instance, delaying their execution. The CFABCast algorithm (Collision-Fast Atomic Broadcast) uses M-Consensus, a Consensus variant, to decide and deliver multiple values in the same instance. However, CFABCast is not Byzantine fault tolerant, a requirement for many systems. Our first contribution is a variation CFABCast that handles Byzantine failures. Unfortunately, the resulting protocol is not collison-fast due to the possibility of Byzantine failures. In fact, our second contribution is the conjecture that there are no Byzantine collision-fast algorithm in the asynchronous model. Finally, our third contribution is a Byzantine collision-fast algorithm that bypasses our impossibility conjecture by using the USIG (Unique Sequential Identifier Generator) trusted component.*

Resumo. *O uso de protocolos de Difusão Atômica baseados em Consenso na implementação de Máquinas de Estados Replicadas esbarra na ineficiência destes protocolos na presença de colisões, isto é, propostas simultâneas. Isso porquê propostas não decididas no Consenso (comandos não entregues) devem ser repropostas em novas instâncias, aumentando seu tempo de entrega e atrasando sua execução. O algoritmo CFABCast (Collision-Fast Atomic Broadcast) utiliza uma variante do Consenso, o M-Consenso, para decidir e entregar múltiplos valores por instância. CFABCast, contudo, não tolera falhas bizantinas, requisito importante em diversos cenários. Como primeira contribuição deste artigo, propomos uma versão modificada do CFABCast que tolera falhas bizantinas. Apesar de ser baseado em um algoritmo rápido à despeito de falhas, nosso algoritmo não é rápido ante a possibilidade de falhas bizantinas. De fato, nossa segunda contribuição é a conjectura de que nenhum algoritmo possa sê-lo no modelo assíncrono. Finalmente, nossa terceira contribuição é a proposta de um algoritmo que é rápido à despeito de falhas bizantinas, contornando a impossibilidade pela extensão do modelo computacional com o componente seguro USIG (Unique Sequential Identifier Generator).*

*Este trabalho foi apoiado pela CAPES no contexto do projeto PVE CAPES 88881.062190/2014-01, pelo CNPq e pela Fapemig.

1. Introdução

Replicação de Máquinas de Estados, ou SMR (*state machine replication*), é uma técnica para obtenção de serviços tolerantes a falhas [Lamport 1978, Lampson 1996]. SMR pode ser implementada por meio de primitivas de Difusão Atômica, ou ABCast (*Atomic Broadcast*), que provê a entrega confiável e ordenada de mensagens a todos os destinatários não faltosos, que executam os comandos contidos nestas mensagens e, consistentemente, modificam seu estado. ABCast, por sua vez, é redutível ao problema de Consenso Distribuído da seguinte forma: sejam infinitas instâncias de Consenso, identificadas univocamente por um inteiro positivo; mensagens a serem difundidas são propostas na primeira instância ainda não decidida e a decisão da i-ésima instância de consenso é a i-ésima mensagem entregue atomicamente. Um problema dessa redução de SMR para Consenso, via ABCast, é que propostas não decididas (comandos não entregues) devem ser repropostas em novas instâncias, aumentando seu tempo de entrega e atrasando sua execução.

Algoritmos que evitam tais reproposições são denominados **rápidos à despeito de colisões** (*collision-fast*) e apresentam latência ótima de dois passos de comunicação. Todos os algoritmos conhecidos desta classe, contudo, possuem certas limitações. Por exemplo, [Mao et al. 2008] não é rápido entre a falha de algum processo e o reinício do algoritmo e [Du et al. 2014] usa relógios sincronizados. Outra importante limitação é que nenhum algoritmo tolera falhas bizantinas, sendo este trabalho o primeiro esforço no sentido de superar tal limitação.

De uma forma geral, este trabalho apresenta três contribuições principais. Primeiro, propomos uma variante do algoritmo CFABCast (*Collision-Fast Atomic Broadcast*) [Schmidt et al. 2014]. Diferentemente do CFABCast, o proposto aqui tolera falhas bizantinas, apesar de em menor número ($f < n/5$ em vez de $f < n/2$), e não é rápido ante a possibilidade de falhas. De fato, nossa segunda contribuição é a conjectura de que nenhum algoritmo possa sê-lo no modelo assíncrono. Finalmente, nossa terceira contribuição é um algoritmo que considera um modelo estendido pelo componente seguro USIG (*Unique Sequential Identifier Generator*) [Veronese et al. 2013], chamado USIG-BCFABCast (*USIG based Byzantine Collision-Fast Atomic Broadcast*). O algoritmo resultante, além de ser rápido à despeito de falhas, tolera $f < n/2$ falhas, mesmo limiar para falhas do tipo *crash* [Schmidt et al. 2014].

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta a fundamentação conceitual para os protocolos propostos, enquanto que os trabalhos relacionados são discutidos na Seção 3. A Seção 4 apresenta o primeiro algoritmo e discute a impossibilidade de um protocolo *collison-fast* no modelo assíncrono. A Seção 5 mostra o protocolo USIG-BCFABCast, que usa o componente seguro USIG para implementar um protocolo *collison-fast* mesmo com a possibilidade de falhas bizantinas. Na Seção 6 são apresentadas nossas considerações finais.

2. Conceitos Fundamentais

2.1. Modelo Computacional

Modelamos problemas de acordo em termos dos papéis desempenhados por agentes nos protocolos, como em [Lamport 1998]. Agentes são entidades que executam alguma tarefa computacional, e.g. processos, *threads*, atores, etc. Por exemplo, no Consenso Distribuído estão presentes os seguintes tipos de agentes:

- **Proposers** (P) propõe valores;
- **Acceptors** (A) cooperam na escolha de um valor como decisão; e
- **Learners** (L) aprendem o valor decidido.

Agentes se comunicam por troca de mensagens em um modelo computacional assíncrono, i.e., no qual não há limite no tempo de transmissão das mensagens ou ações executadas pelos agentes. Um agente é correto se não é falho, e é falho se sua execução foge da especificação do protocolo, podendo inclusive executar ações arbitrárias e maliciosas (bizantinas). Mensagens podem ser perdidas ou duplicadas mas não indetectavelmente corrompidas, e se repetidamente reenviadas de um agente correto para outro, são entregues em algum momento.

Consideramos também que cada agente possui um par de chaves (pública e privada) usadas para gerar e verificar assinaturas digitais; todo agente conhece apenas sua própria chave privada, mas todas as chaves públicas. Somente mensagens corretamente assinadas são processadas nos receptores. Representamos uma mensagem m assinada por uma agente x como $\langle m \rangle_{\sigma_x}$.

2.2. M-Consensus

O algoritmo Collision-Fast Atomic Broadcast (CFABcast) usa instâncias do algoritmo Collision-Fast Paxos, que resolve o problema M-Consenso. Nesta variante do Consenso Distribuído, agentes devem concordar em um mapeamento de *proposers* para valores propostos ou um valor especial *Nil*[Schmidt et al. 2014]. Esse mapeamento é capturado pela estrutura de dados *v-mapping*, i.e. mapa de valores, ou v-map, que contêm um conjunto, potencialmente vazio, de funções que mapeiam *proposers* a valores propostos ou *Nil*.

Os mapas possuem uma relação de precedência, de forma que se pode estabelecer uma ordem total entre eles (e.g. usando um identificador único para cada *proposer*). Um v-map com apenas um elemento em seu Domínio (i.e. conjunto de *proposers*) é denominado simples ou um *single-mapping*. Um *v-mapping* é dito trivial se todos os elementos de seu domínio mapeiam para o valor *Nil*. Finalmente, dois *v-mapping* são compatíveis se os elementos na intersecção de seus domínios mapeiam para os mesmos valores. Em outras palavras, estes mapas podem ser estendidos até que se tornem iguais, pois não há discordância entre eles. Assim, em um algoritmo de M-Consenso, *proposers* propõem valores e *learners* aprendem v-map, possivelmente diferentes mas sempre compatíveis, e que são estendidos por mapas triviais até que sejam iguais.

Formalmente, o M-Consenso é definido pelas seguintes propriedades, onde $learned[l]$ é o valor aprendido pelo $learner\ l \in L$, inicialmente $\perp$ (vazio), e $s \sqsubseteq r$ significa que s é prefixo de r e que r é uma extensão de s:

- *Não trivialidade:* $\forall l \in L, learned[l] = \perp$, $learned[l]$ é sempre um v-map proposto e não-trivial.
- *Estabilidade:* $\forall l \in L$, se $learned[l] = s$ em algum momento, então $s \sqsubseteq learned[l]$ em todos momentos posteriores.
- *Consistência:* O conjunto de v-map aprendidos é sempre compatível e não-trivial.

O objetivo do M-Consenso é fazer com que todos os *learners* aprendam, em algum momento, um *v-map* completo, isto é, um *v-map* em que todos os agentes que propuseram valores, são mapeados para os valores propostos ou para Nil. Isso só é possível se existir

um quorum de *acceptors* funcional durante a execução do algoritmo, sendo que um subconjunto dos *acceptors* forma um quorum se tem interseção não vazia com qualquer outro quorum. Formalmente, o progresso no M-Consenso é dado pela seguinte propriedade:

- *Progresso:* Para qualquer $p \in P$ e $l \in L$, se p, l e um quorum de *acceptors* não são falhos e p propõe um valor, então $learned[l]$ se tornará completo em algum momento.

2.3. Difusão Atômica

Difusão Atômica é definida pelas seguintes propriedades, onde $delivered[l]$ é a sequência de mensagens entregues pelo *learner* $l \in L$, inicialmente vazia e $\sqsubseteq$ é o operador de prefixo entre sequências:

- *Não trivialidade:* $\forall l \in L, delivered[l]$ contém apenas mensagens difundidas e não duplicadas.
- *Estabilidade:* $\forall l \in L$, se $delivered[l] = s$ em algum momento, então $s \sqsubseteq delivered[l]$ em todos momentos posteriores.
- *Consistência:* $\forall l_1, l_2 \in L$, ou $delivered[l_1] \sqsubseteq delivered[l_2]$ ou $delivered[l_2] \sqsubseteq delivered[l_1]$.

Como mostrado em [Schmidt et al. 2014], Difusão Atômica pode ser reduzido ao problema de M-Consensus de forma semelhante à redução ao Consenso, com a vantagem de entregar múltiplas mensagens por instância de M-Consensus em vez de apenas uma.

3. Trabalhos Relacionados

Esta seção descreve os trabalhos relacionados com os protocolos aqui propostos. Primeiramente, os algoritmos do Paxos e algumas variantes são discutidos e as principais diferenças entre estes protocolos são destacadas. Por fim, o algoritmo *Collision-fast Atomic Broadcast* (CFABCast) é apresentado, pois o mesmo deve ser instanciado com os protocolos propostos nas seções seguintes para a concretização das variantes bizantinas: BCFABCast e USIG-BCFABCast.

3.1. Paxos e Variantes

Paxos é um protocolo de consenso [Lamport 1998] notório por seu uso na indústria, e.g., *Chubby Lock Service* [Burrows 2006]. No *Paxos*, as propostas (comandos) são enviadas pelos clientes para um *proposer* eleito *coordenador* para que as repasse aos *acceptors* para serem decididas e entregues. Desde que não haja mais de um processo que se julgue coordenador e que exista um quorum de *acceptors* corretos e alcançáveis, a instância decidirá.

O algoritmo é executado em rodadas, e cada rodada possui duas fases, com dois passos de comunicação cada; a fase 1 é de preparação do protocolo e a fase 2 é que efetivamente leva a uma decisão. Em um cenário em que múltiplas instâncias são necessárias, a fase 1 pode ser executada em paralelo para todas as instâncias, amortizando seu custo, e reduzindo a decisão à execução da fase 2. Assim, se o coordenador faz uma proposta, o valor pode ser decidido em dois passos de comunicação. Para os demais *proposers*, 3 passos são necessários.

Fast Paxos [Lamport 2006a] é um algoritmo que "contorna" o coordenador para reduzir o tempo de decisão para dois passos para múltiplos *proposers*; algoritmos similares são conhecidos em geral como rápidos (*fast*). Contudo, contornar o coordenador pode levar o algoritmo a não progredir na presença de colisões, isto é, propostas concorrentes para a mesma instância. Algoritmos *Collision-Fast* [Lamport 2006b] decidem mesmo na presença de colisões mas, como todo algoritmo de Consenso, decidem por apenas um valor ao final de cada instância. O mesmo é válido para versões bizantinas destes protocolos, como Bizantine Paxos (BFT Paxos)[Castro and Liskov 2002] e Bizantine Fast Paxos (FAB)[Martin and Alvisi 2006]. Assim, na redução clássica de Difusão Atômica para Consenso Distribuído, mensagens propostas mas não decididas em uma instância precisam ser repropostas, aumentando a sua latência. CFABcast [Schmidt et al. 2014], discutido a seguir, contorna este problema permitindo que múltiplos valores sejam decididos por instância. Até agora, contudo, este protocolo não tinha uma versão que tolerasse falhas bizantinas.

3.2. Collision-fast Atomic Broadcast

Em [Schmidt et al. 2014] os autores demonstram a redução da Difusão Atômica ao problema de M-Consensus. Em específico, apresentam o algoritmo *Collision-fast Atomic Broadcast* (CFABCast), reproduzido aqui como Algorithm 1. O CFAbcast utiliza infinitas instâncias de *Collision-fast Paxos* (CFPaxos), que resolve M-Consensus, para entregar diversas mensagens em paralelo. As ações do CFABCast são diretamente mapeadas em ações do CFPaxos e, de forma simplificada, consistem em escolher a instância CFPaxos com menor identificador e que ainda não tenha decidido ou proposto nesta instância a mensagem a ser entregue. Para mais detalhes, recomendamos a leitura de [Schmidt et al. 2014], que inclui detalhada prova de corretude. É importante notar que para aumentar as chances de terminação no CFPaxos, somente um subconjunto dos *proposers*, os *collision-fast proposers* ou CF-proposers, pode propor em determinada rodada. Este subconjunto é determinado pelo coordenador da rodada, seguindo algum oráculo que monitora o sistema [Saramago 2016].

Algorithm 1 *Collision-fast Atomic Broadcast [Schmidt et al. 2014]*

I: the set of all Collision-fast Paxos instances used
$CFP(i)!A$: the action or variable A of Collision-fast Paxos instance i

1: $Propose(p, V) \triangleq$
2: $\quad \forall i \in I, CFP(i)!Propose(p, V)$
3: $NewPhase1a(i, c, r) \triangleq$
4: **pre-conditions:**
5: $\quad c = C(r)$
6: $\quad crnd[c] < r$
7: $\quad c$ believes itself to be the leader
8: $\quad c$ heard of a round $r > j > crnd[c]$ for some instance or $CF(crnd[c]) \notin active[c]$
9: **actions:**
10: $\quad CFP(i)!Phase1a(c, r)$
11: $Phase1a(c, r) \triangleq$
12: $\quad \forall i \in I, CFP(i)!NewPhase1a(i, c, r)$
13: $Phase1b(a, r) \triangleq$
14: $\quad \forall i \in I, CFP(i)!Phase1b(a, r)$
15: $Phase2Start(c, r) \triangleq$
16: $\quad \forall i \in I, CFP(i)!Phase2Start(c, r)$
17: $Phase2Prepare(p, r) \triangleq$
18: $\quad \forall i \in I, CFP(i)!Phase2Prepare(p, r)$
19: $Phase2a(p, r, V) \triangleq$
20: **pre-condition:**
21: $\quad p$ has not yet proposed V
22: **action:**
23: $\quad$ LET $i = Min([j : CFP(j)!pval[p] = none])$
24: $\quad CFP(i)!Phase2a(p, r, V)$
25: $Phase2b(i, a, r) \triangleq$
26: $\quad CFP(i)!Phase2b(a, r)$
27: $Learn(i, l) \triangleq$
28: $\quad CFP(i)!Learn(l)$

Enquanto [Schmidt et al. 2014] consideravam apenas falhas do tipo *crash*, neste trabalho estamos interessados em falhas bizantinas. Assim, apresentamos duas variantes

do CFPaxos que toleram tais falhas e que podem ser usadas sem adaptações pelo CFAB-Cast. A primeira variante é denominada BCFABCast e é o primeiro algoritmo bizantino para o M-Consensus; apesar de ser baseada no CFPaxos, não é *collision-fast* e é apresentada como um passo intermediário para o nosso algoritmo final, USIG-BCFABCast.

3.3. Resumo

Para facilitar a comparação dos algoritmos discutidos aqui, a Tabela 1 apresenta várias métricas dos mesmos, sendo que as duas últimas colunas referem-se aos protocolos que estamos propondo.

	Paxos	Fast Paxos	CFABCast	BFT Paxos	BCFABCast	USIG-BCFABCast
#passos	3	2	2	4	3	2
#acceptors	2f + 1	3f + 1	2f + 1	3f + 1	5f + 1	2f + 1
#quorum	f + 1	2f + 1	f + 1	2f + 1	4f + 1	f + 1
Falha	Crash	Crash	Crash	Bizantinas	Bizantinas	Bizantinas
Componente	–	–	–	–	–	USIG

Tabela 1. Para cada protocolo, considerando até f falhas: número de passos de comunicação para se alcançar acordo; número de réplicas desempenhando o papel de *acceptors* necessárias para garantir corretude; tamanho dos quóruns de *acceptors*; tipo de falha suportada; e necessidade de um componente seguro.

4. *Byzantine* Collision-Fast Paxos

No Algorithm 2 introduzimos o Byzantine Collision-fast Paxos, que resolve M-Consenso na presença de falhas bizantinas. Quando usado em conjunto com o Algoritmo 1, resolve o problema de Difusão Atômica na presença de falhas bizantinas. Contudo, apesar do nome e de ser derivado do *Collision-Fast Paxos*, o protocolo não é rápido à despeito de colisões. Na Seção 5 sanaremos esta deficiência.

4.1. Visão Geral

Como outras variantes do Paxos, cada instância do nosso algoritmo é executada em rodadas divididas em duas fases. Na fase 1 o coordenador questiona os *acceptors* (ação $Phase1a$) por valores aceitos em rodadas anteriores. Baseado nas respostas dos *acceptors* (emitidas na ação $Phase1b$), o coordenador estabelece se algum v-map já foi possivelmente decidido (ação $Phase2Start$).

Observe que o protocolo garante que se algum mapeamento tiver sido decidido então pelo menos $2f + 1$ mensagens $1b$ conterão tal mapeamento. Seja S o conjunto de todos os *single-maps* aceitos por $2f + 1$ acceptors. Se S não é vazio, o coordenador informa os CF-proposers, para que se abstenham de propor nesta rodada, e os *acceptors*, para que aceitem os mapeamentos em S e informem os *learners* (ação $Phase2b$). Assim, o coordenador tenta terminar a instância e progredir na computação.

Caso contrário, o coordenador informa os *CF-proposers* da rodada sendo iniciada, para que proponham seus valores para os *acceptors* (ação $Phase2Prepare$). Autorizado pelo coordenador, um *CF-proposer* pode repassar o valor em mensagem $\langle$“`propose`”, _$\rangle$ recebida de um *proposer* qualquer, incluindo si mesmo (ação $Phase2a$). *CF-proposers* repassam a mensagem $2S$ recebida do coordenador para os *acceptors* para provarem que tem permissão de propor.

Algorithm 2 *Byzantine Collision-fast Paxos*

Pr, A, L: proposers, acceptors and learners sets;
$CF(i)$: round i's collision-fast proposers set;
$C(i)$: round i's coordinator.
$prnd[p]$, $crnd[c]$, $rnd[a]$: current round of proposer p, coordinator c, and acceptor a, respectively, initially 0.
$pval[p]$: value p has fast-proposed at $prnd[p]$ or $none$ if p has not fast-proposed at $prnd[p]$, initially $none$.
$cval[c]$: initial v-mapping for $crnd[c]$, if c has queried an acceptor quorum or $none$ otherwise; initially $\perp$ for coordinator of round 0 and $none$ for others.
$vrnd[a]$: round at which a has accepted its latest value.
$vval[a]$: v-mapping a has accepted at $vrnd[a]$ or $none$ if no value accepted at $vrnd[a]$; initially $none$.
$learned[l]$: v-mapping currently learned by learner l; initially $\perp$.
$m \Leftarrow s$: received message m from source s
$m \Rightarrow d$: send message m to destination d

1: $Propose(p, V) \triangleq$
2: **pre-condition:**
3: $p \in Pr$
4: **action:**
5: $\langle$"propose", $V\rangle_{\sigma_p} \Rightarrow cf \in CF(prnd[p])$
6: $Phase1a(c, r) \triangleq$
7: **pre-conditions:**
8: $c = C(r)$
9: $crnd[c] < r$
10: **actions:**
11: $crnd[c] \leftarrow r$
12: $cval[c] \leftarrow none$
13: $\langle$"1a", $r\rangle_{\sigma_c} \Rightarrow A$
14: $Phase1b(a, r) \triangleq$
15: **pre-conditions:**
16: $a \in A$
17: $rnd[a] < r$
18: $\langle$"1a", $r\rangle_{\sigma_c} \Leftarrow C(r)$
19: **actions:**
20: $rnd[a] \leftarrow r$
21: $\langle$"1b", $r, vrnd[a], vval[a]\rangle_{\sigma_a} \Rightarrow C(r)$
22: $Phase2Start(c, r) \triangleq$
23: **pre-conditions:**
24: $c = C(r)$
25: $crnd[c] = r$
26: $cval[c] = none$
27: $\exists Q \subseteq A$:
28: Q is a quorum
29: $\forall a \in Q, \langle$"1b", $r, vrnd, vval\rangle_{\sigma_a} \Leftarrow a$
30: **actions:**
31: LET $1bs = [\ m = \langle$"1b", _, _, _$\rangle_{\sigma_a} : m \Leftarrow a \in Q\ ]$
32: LET $k = Max([vrnd : \langle$"1b", _, $vrnd$, _$\rangle_{\sigma_a} \in 1bs\ ])$
33: LET $A_{\langle p,v\rangle} = [a : \langle$"1b", $r, k, vval\rangle_{\sigma_a} \in 1bs \wedge vval[p] = v]$
34: LET $S = \left[\langle p, v\rangle : A_{\langle p,v\rangle} \geq 2f + 1\right]$
35: IF $S = \emptyset$ THEN
36: $cval[c] \leftarrow \perp$
37: $\langle$"2S", $r, cval[c], msgs\rangle_{\sigma_c} \Rightarrow CF(r)$
38: ELSE
39: $cval[c] \leftarrow \sqcup S \bullet [\langle p, Nil\rangle : p \in CF(r)]$
40: $\langle$"2S", $r, cval[c], msgs\rangle_{\sigma_c} \Rightarrow CF(r) \cup A$
41: $Phase2Prepare(p, r) \triangleq$
42: **pre-conditions:**
43: $p \in CF(r)$
44: $prnd[p] < r$
45: $\langle$"2S", $r, v, proofs\rangle_{\sigma_{C(r)}} \Leftarrow C(r)$
46: goodRoundValue(r,v,proofs)
47: **actions:**
48: $prnd[p] \leftarrow r$
49: $proof[p] \leftarrow proofs$
50: IF $v = \perp$ THEN $pval[p] \leftarrow none$
51: ELSE $pval[p] \leftarrow v(p)$
52: $Phase2a(p, r, V) \triangleq$
53: **pre-conditions:**
54: $p \in CF(r)$
55: $prnd[p] = r$
56: $pval[p] = none$
57: **either** $(V \neq Nil \wedge \langle$"propose", $V\rangle_{\sigma_p} \Leftarrow p \in Pr)$
58: **or** $(V = Nil \wedge \langle$"2a", $r, \langle q, W\rangle\rangle_{\sigma_q} \Leftarrow q \in CF(r) \wedge W \neq Nil)$
59: **actions:**
60: $pval[p] \leftarrow V$
61: IF $V \neq Nil$ THEN
62: $\langle$"2a", $r, \langle p, V\rangle, proof[p]\rangle_{\sigma_p} \Rightarrow A \cup CF(r)$
63: ELSE
64: $\langle$"2a", $r, \langle p, V\rangle, proof[p]\rangle_{\sigma_p} \Rightarrow A$
65: $Phase2b(a, r) \triangleq$
66: LET $Cond1 =$
67: $vval[a] = none \vee$
68: $(\langle$"2S", $r, v, proofs\rangle_{\sigma_c} \Leftarrow C(r) \wedge$
69: goodRoundValue(r, v, proofs)
70: $v \neq \perp \wedge vrnd[a] < r)$
71: LET $Cond2 =$
72: $\langle$"2a", $r, \langle p, V\rangle, proofs\rangle_{\sigma_p} \Leftarrow p \in CF(r) \wedge V \neq Nil$
73: goodRoundValue(r, V, proofs)
74: **pre-conditions:**
75: $a \in A$
76: $rnd[a] \leq r$
77: $Cond1 \vee Cond2$
78: **actions:**
79: IF $Cond1$
80: THEN $vval[a] \leftarrow v$
81: ELSE
82: IF $Cond2 \wedge (vrnd[a] < r \vee vval[a] = none)$
83: THEN $vval[a] \leftarrow \perp \bullet \langle p, V\rangle \bullet [\langle p, Nil\rangle : p \in Pr \setminus CF(r)]$
84: ELSE $vval[a] \leftarrow vval[a] \bullet \langle p, V\rangle$
85: $rnd[a] \leftarrow vrnd[a] \leftarrow r$
86: $\langle$"2b", $r, vval[a]\rangle_{\sigma_a} \Rightarrow L$
87: $Learn(l) \triangleq$
88: **pre-conditions:**
89: $l \in learners$
90: $\exists Q \subseteq A$:
91: Q is a quorum
92: $\forall a \in Q, \langle$"2b", $r, _\rangle_{\sigma_a} \Leftarrow a$
93: **actions:**
94: $Q2bVals = \{v : \langle$"2b", $r, v\rangle_{\sigma_a} \Leftarrow a \in Q\}$
95: $w = \sqcap Q2bVals$
96: $learned[l] = learned[l] \sqcup w$

Para que um coordenador bizantino não possa ignorar as propostas já aceitas, propondo qualquer valor na nova rodada e violando a propriedade de acordo do consenso, o coordenador prova que executou a fase 1 corretamente. Isto é feito enviando-se o conjunto das mensagens recebidas de um quórum de *acceptors* e usadas para computar S (linhas

31-35) nas mensagens $2S$. Os agentes receptores fazem o mesmo cálculo para verificar se o valor proposto realmente é válido para a rodada em questão (função $goodRoundValue$).

Ao receberem propostas e verificarem sua validade (ação $Phase2b$), *acceptors* estendem os v-maps que já tenham aceito e repassam para os *learners*. *Learners*, por sua vez, coletam, dos v-maps recebidos dos *acceptors*, os *single-maps* aceitos por um quorum (linha 93), e os agregam ao v-map aprendido (ação $Learn$).

4.2. Corretude

Em alto nível, a corretude do BCFABCast é herdada do CFABCast, pois a execução do primeiro imita a execução do segundo, com alguns passos extra para coibir agentes de agir maliciosamente. Estas diferenças se concentram nos *acceptors* e *CF-proposers*. Isto porquê *learners* bizantinos não podem atrapalhar outros agentes, uma vez que não enviam mensagens. Contudo, podem aprender qualquer valor, mas, até onde sabemos, este problema é incontornável. Quanto a *proposers*, estes são normalmente clientes do sistema e é difícil impedir que sejam comprometidos. Contudo, mesmo um *proposer* malicioso não pode deixar de seguir o protocolo, pois a única ação que executa é propor valores. Neste trabalho desconsideramos ataques de negação de serviço, que podem ser tratados usando técnicas específicas descritas na literatura.

De um total de $5f$ acceptors, $4f + 1$ devem ser corretos. Assim, de um quórum de $4f+1$, garantidamente $3f+1$ serão corretos. *learners* esperam por $4f+1$ confirmações do mesmo mapeamento antes de aprendê-lo, aprendendo apenas valores aceitos por um quorum e também detectar caso um agente se comporte de forma inconsistente. Na pior das hipóteses, acceptors e *CF-Proposer* bizantinos podem entrar em conluio para aceitarem diferentes valores, repassados a *learners* diferentes. Embora este ataque possa impedir uma rodada de terminar, ela não pode levar a decisões conflitantes.

Um coordenador correto que inicie uma nova rodada também pode, em virtude dos quóruns de $4f + 1$ *acceptors*, identificar qualquer mapeamento possivelmente decidido, uma vez que o mesmo deve ter sido aceito por pelo menos $2f + 1$ dos *acceptors* que responderem com mensagens $1b$, pois no máximo f *acceptors* bizantinos e f *acceptors* corretos podem reportar outros valores.

Se o coordenador for bizantino, este poderia ignorar as mensagens recebidas dos *acceptors* durante a fase 1 na tentativa de violar a propriedade de acordo do protocolo. Por isso o protocolo exige que o coordenador prove que executou a fase 1 corretamente. A prova é constituída das mensagens $\langle$"`1b`", $a\rangle$ssinadas, previamente recebidas de um quorum, que o coordenador não conseguiria forjar. Logo, um coordenador não consegue manipular o que já foi aprendido sem ser detectado, embora possa forçar reconfigurações corretas indefinidamente, comprometendo o progresso do protocolo. Este último problema pode ser resolvido utilizando-se, por exemplo, o mecanismo de Multi-Coordenação proposto em [Camargos et al. 2007] ou o rodízio do papel do coordenador, limitando seu ataque.

4.3. Impossibilidade de *Byzantine Collision-fast Atomic Broadcast*

No CFABCast [Schmidt et al. 2014], se um *CF-proposer* i não tem um valor a propor e recebe a proposta de outro *CF-proposer* j, i envia uma mensagem diretamente aos *learners*, avisando que se absterá de propor nesta rodada. *learner*, ao receber a mensagem de

i, aprende que i não poderá ser mapeado a outro valor, não precisando do aceite dos *acceptors*. Este cenário é apresentado na Figura 1, à esquerda, em que há dois *Collision-fast proposer* CFP_0 e CFP_1, e nenhuma falha, bizantina ou não, acontece. Assim, o protocolo consegue terminar em dois passos de comunicação sempre que todas as propostas forem feitas concorrentemente.

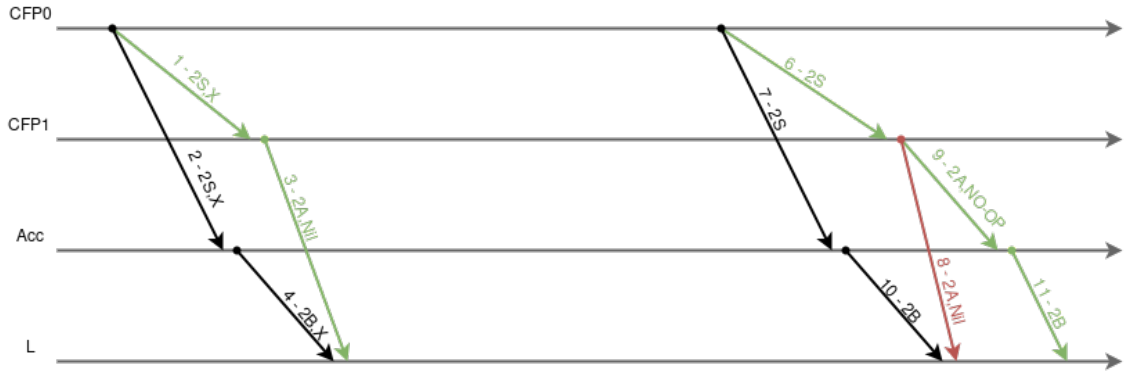


Figura 1. Quantidade de passos de comunicação necessários para se decidir um *v-mapping* completo no CFPaxos (esquerda) e BCFPaxos (direita).

Se a mesma abordagem fosse possível no BCFABCast, então seria possível que um *CF-Proposer* bizantino dissesse que está se abstendo ao mesmo tempo que envia uma proposta aos *acceptors*. Este cenário é apresentado na Figura 1, do lado direito, onde o CFP_0 propôs algum valor e CFP_1 propôs *Nil* direto para o *learner*, Para evitar esse caso, no BCFABCast não há contato direto entre *CF-Proposer* e *learner* e mesmo as abstenções devem ser confirmadas por um quorum antes de se tornarem parte do v-map aprendido.

Para que um *CF-proposer* não possa propor *Nil* continuamente, impedindo o progresso do protocolo, exigimos que o mesmo só proponha caso tenha recebido mensagem "$propose''$ ou "$2a$". Assim, caso um *CF-Proposer* não tenha nada a propor, serão necessários até três passos de comunicação para terminar o protocolo: 1 passo para receber a mensagem "$2a$" e mais 2 passos para enviar a abstenção aos *learners* via os acceptors.

Nos parece que tal restrição seja válida a qualquer protocolo e , assim, conjecturamos que nenhum protocolo bizantino possa ser *collision fast*, ou seja, decidir em dois passos de comunicação na possível presença de agentes bizantinos. Fazê-lo implicaria em confiar nas mensagens recebidas diretamente de um *CP-proposer*, talvez bizantino.

5. Estendendo o modelo de sistema com o uso de um componente seguro

Nesta seção mostramos que adicionando um componente seguro no sistema é possível reduzir o número tanto de réplicas (de $5f + 1$ para $2f + 1$ acceptors) quanto de passos de comunicação (de 3 para 2) necessários para resolver o BCFABCast. Particularmente interessante, mostramos que é possível resolver o BCFABCast com apenas 2 passos de comunicação, mesmo número necessário para resolver o CFABCast [Schmidt et al. 2014].

Os componentes seguros propostos para auxiliar no desenvolvimento de protocolos que tolerem falhas maliciosas possuem diferenças consideráveis tanto em aspectos de implementação quanto de desempenho. Quanto à implementação, estes componentes podem ser: i) implementados de forma distribuída, como o *Trusted Timely Computing Base*

(TTCB) [Correia et al. 2004]; ii) locais baseados em memória, como o *Attested Append-Only Memory* (A2M) [Chun et al. 2007]; e iii) locais baseados em um simples contador, como o *Unique Sequential Identifier Generator* (USIG) [Veronese et al. 2013].

Basicamente, um atacante não consegue acessar estes componentes, mesmo que consiga invadir um servidor. Com isso, é possível projetar protocolos que restringem as ações que um processo malicioso (invadido) pode executar sem ser descoberto. Neste trabalho escolhemos o componente seguro USIG [Veronese et al. 2013] pela sua simplicidade, além da fácil implementação e utilização no sistema.

5.1. USIG: Gerador de identificadores sequenciais únicos

Este componente é local em cada processo (proposer, acceptor ou learner) e será usado para atribuir um identificador único para uma mensagem, além de assiná-la. Para cada processo, os identificadores atribuídos às mensagens são únicos (um mesmo identificador nunca é atribuído a duas ou mais mensagens), monotônicos (o identificar atribuído a uma mensagem nunca é menor do que o anterior) e sequenciais (o identificador atribuído a uma mensagem sempre é o sucessor do anterior).

A interface definida para acessar este serviço é a seguinte [Veronese et al. 2013]:

- `createUI(m)`: retorna um certificado que contém um identificador único `UI` e a prova de que este identificador foi criado por este componente e atribuído a `m`. O identificador é basicamente um contador monotonicamente crescente que é incrementado a cada chamada desta função. Já a prova envolve criptografia e pode ser baseada em um *hash* (*Hash-based Message Authentication Code* – HMAC) ou em criptografia assimétrica (assinaturas digitais).
- `verifyUI(PK,UI,m)`: verifica se o identificador único `UI` é válido para `m`.

Com o uso deste componente, um processo não consegue enviar duas mensagens diferentes para processos diferentes com um mesmo identificador. Assim, basta que cada processo mantenha armazenado o identificador da última mensagem recebida de um processo para saber qual é o próximo identificador esperado, e assim reduz-se as ações de um processo malicioso: ou envia a mesma mensagem (que pode conter qualquer valor) para todos os processos ou não envia nada.

Este componente pode ser implementado de duas formas [Veronese et al. 2013]: é possível usar apenas *hashes* ou empregar assinaturas digitais. Neste trabalho adotaremos a solução que emprega assinaturas digitais, de forma que a verificação pode ser realizada fora do componente seguro, bastando a chave pública correspondente. Além disso, diferentes níveis de isolação podem ser empregados, usando máquinas virtuais ou até mesmo módulos seguros de hardware (*Trusted Platform Module* – TPM).

5.2. USIG-BCFABCast: USIG *based Byzantine Collision-fast Atomic Broadcast*

Esta seção apresenta o protocolo para resolver o BCFABCast usando o componente seguro USIG, chamado de USIG-BCFABCast (*USIG based Byzantine Collision-fast Atomic Broadcast*). Mais especificamente, o Algoritmo 3 substitui o Collision-Fast Paxos na resolução do M-Consensus e deve ser usado em conjunto com o Algoritmo 1. Devido ao uso deste componente seguro, o protocolo proposto necessita de apenas $2f + 1$ *acceptors* e 2 passos de comunicação, sendo rápido à despeito de colisões mesmo na presença de processos maliciosos.

A ideia geral é que através dos identificadores únicos adicionados às mensagens, os *collision-fast proposers* não precisam enviar suas mensagens através dos *acceptors* caso não tenham nada a propor, i.e., os *learners* podem confiar nas propostas, com valores nulos, recebidas diretamente dos *proposers* uma vez que eles não conseguem enviar propostas diferentes com o mesmo identificador. Note que as propostas com valores continuam seguindo o caminho normal através dos *acceptors*, visto que caso o coordenador seja trocado (devido a falhas ou assincronismos) estes valores estão armazenados nos *acceptors* e são usados para configuração do novo *round* como no algoritmo anterior (Seção 4).

5.2.1. Visão Geral

O protocolo proposto é apresentado no Algoritmo 3. As fases iniciais de configuração de um novo *round* pelo novo coordenador não precisam de grandes alterações, pois caso um coordenador envie diferentes valores para o *round r* nas mensagens $1a$, o mesmo não vai conseguir uma prova para reconfigurar o sistema com as mensagens $2S$. De fato, a primeira alteração ocorre nas mensagens $2S$, que carrega um identificador único gerado pelo USIG do coordenador e impossibilita que mensagens diferentes sejam enviadas a agentes (processos) diferentes. Consequentemente, todos os agentes recebem a mesma configuração inicial para o *round r*.

Cada *CF-proposer* também adiciona um identificador único às suas propostas nas mensagens $2a$, impossibilitando que propostas diferentes sejam enviadas para agentes diferentes. Apesar destas mensagens serem para todos os agentes ($A \cup CF(r) \cup L$), apenas os *learners* processam as propostas com Nil (linhas 46,60 e 86). Isso é necessário para que os *acceptors* e os outros *CF-Proposer* possam incrementar seus contadores para as propostas esperadas mesmo quando a proposta é Nil. Por outro lado, os *learners* incrementam seus contadores para propostas diferentes de Nil quando recebem estas propostas dos *acceptors* (linha 84). Note também que os identificadores únicos sempre acompanham suas respectivas propostas, de forma que quando uma nova rodada precisa ser iniciada, os mesmos são enviados pelos *acceptors* para o novo coordenador, que os encaminha nas mensagens $2S$ para então serem repassados pelos *acceptors*, aos *learners* nas mensagens $2b$ (linhas 69 e 76).

Finalmente, vale destacar que os *acceptors* também adicionam um identificador único às mensagens $2b$ enviadas aos *learners* (linhas 75–76). Consequentemente, um *acceptor* malicioso é incapaz de enviar diferentes mensagens para diferentes *learners* sem ser descoberto. De fato, os *learners* apenas processam a mensagem com o contador imediatamente seguinte na sequência (linha 84).

5.2.2. Corretude

Através do uso de identificadores sequenciais únicos, o protocolo proposto no Algoritmo 3 impede que um agente malicioso (coordenador, *acceptor* ou *proposer*) envie mensagens diferentes para diferentes agentes em um mesmo passo do algoritmo. Isto torna o funcionamento do protocolo semelhante ao [Schmidt et al. 2014], que tolera apenas falhas por *crash*: ou um agente envia um mesmo conteúdo (que pode ser forjado) para todos

Algorithm 3 USIG *based Byzantine Collision-fast Paxos*

All variables and sets from Algorithm 2
$USIG^C[c]$, $cnt[c]$: USIG used by coordinator c and expected counter value for msgs received from c (initially, 0)
$USIG^A[a]$, $cnt[a]$: USIG used by acceptor a and expected counter value for msgs received from a (initially, 0)
$USIG^P[p]$, $cnt[p]$: USIG used by CF proposer p and expected counter value for msgs received from p (initially, 0)
verifyCnt(UI_x,$cnt[x]$) ≡ **if**($UI_x.cnt = cnt[x]$)**then**$cnt[x]$++; *return true*; **else** *return false*;

1: $Propose(p, V) \triangleq$
2: lines 2 – 5 of Algorithm 2

3: $Phase1a(c, r) \triangleq$
4: lines 7 – 13 of Algorithm 2

5: $Phase1b(a, r) \triangleq$
6: lines 15 – 21 of Algorithm 2

7: $Phase2Start(c, r) \triangleq$
8: **pre-conditions:**
9: $c = C(r)$
10: $crnd[c] = r$
11: $cval[c] = none$
12: $\exists Q \subseteq A$:
13: Q is a quorum
14: $\forall a \in Q, \langle$"1b", $r, vrnd, vval\rangle_{\sigma_a} \Leftarrow a$
15: **actions:**
16: LET $msgs = [\ m = \langle$"1b", $r, vrnd, vval\rangle_{\sigma_a} : m \Leftarrow a \in Q\ ]$
17: LET $k = Max([vrnd : \langle$"1b", $r, vrnd, vval\rangle_{\sigma_a} \in msgs\ ])$
18: LET $S = [vval : \langle$"1b", $r, k, vval\rangle_{\sigma_a} \in msgs, vval \neq none]$
19: IF $S = \emptyset$ THEN
20: $cval[c] \leftarrow \perp$
21: ELSE
22: $cval[c] \leftarrow \sqcup S \bullet [\langle p, Nil, Nil\rangle : p \in CF(r)]$
23: $UI_c \leftarrow USIG^C[c]$.createUI($\langle$"2S", $r, cval[c], msgs\rangle$)
24: $\langle$"2S", $r, cval[c], msgs, UI_c\rangle \Rightarrow CF(r) \cup A$

25: $Phase2Prepare(p, r) \triangleq$
26: **pre-conditions:**
27: $p \in CF(r)$
28: $prnd[p] < r$
29: $\langle$"2S", $r, v, proof, UI_{C(r)}\rangle \Leftarrow C(r)$
30: goodRoundValue(r,v,proofs)
31: verifyUI($PK_{C(r)}$,$UI_{C(r)}$,$\langle$"2S", $r, v, proofs\rangle$)
32: verifyCnt($UI_{C(r)}$,$cnt[C(r)]$)
33: **actions:**
34: $prnd[p] \leftarrow r$
35: IF $v = \perp$ THEN $pval[p] \leftarrow none$
36: ELSE $pval[p] \leftarrow v(p)$

37: $Phase2a(p, r, V) \triangleq$
38: **pre-conditions:**
39: $p \in CF(r)$
40: $prnd[p] = r$
41: $pval[p] = none$
42: **either** ($V \neq Nil \wedge \langle$"propose", $V\rangle_{\sigma_p} \Leftarrow p \in Pr$)
43: **or** ($V = Nil \wedge$
44: $\langle$"2a", $r, \langle q, W\rangle, UI_q\rangle \Leftarrow q \in CF(r) \wedge$
45: verifyUI(PK_q,UI_q,$\langle q, W\rangle$) $\wedge$
46: verifyCnt(UI_q,$cnt[q]$) $\wedge W \neq Nil$)
47: **actions:**
48: $pval[p] \leftarrow V$
49: $UI_p \leftarrow USIG^P[p]$.createUI($\langle p, V\rangle$)
50: $\langle$"2a", $r, \langle p, V\rangle, UI_p\rangle \Rightarrow A \cup CF(r) \cup L$

51: $Phase2b(a, r) \triangleq$
52: LET $Cond1 =$
53: $vval[a] = none \vee$
54: ($\langle$"2S", $r, v, proofs, UI_{C(r)}\rangle \Leftarrow C(r) \wedge$
55: verifyUI($PK_{C(r)}$,$UI_{C(r)}$,$\langle$"2S", $r, v, proofs\rangle$) $\wedge$
56: verifyCnt($UI_{C(r)}$,$cnt[C(r)]$) $\wedge$
57: goodRoundValue(r,v,proofs) $\wedge$
58: $\wedge v \neq \perp \wedge vrnd[a] < r$)
59: LET $Cond2 =$
60: $\langle$"2a", $r, \langle p, V\rangle, UI_p\rangle \Leftarrow p \in CF(r) \wedge V \neq Nil \wedge$
61: verifyUI(PK_p,UI_p,$\langle p, V\rangle$) $\wedge$
62: verifyCnt(UI_p,$cnt[p]$)
63: **pre-conditions:**
64: $a \in A$
65: $rnd[a] \leq r$
66: $Cond1 \vee Cond2$
67: **actions:**
68: IF $Cond1$
69: THEN $vval[a] \leftarrow v$
70: ELSE
71: IF $Cond2 \wedge (vrnd[a] < r \vee vval[a] = none)$
72: THEN $vval[a] \leftarrow \perp \bullet \langle p, V, UI_p\rangle \bullet [\langle p, Nil, Nil\rangle : p \in Pr \setminus CF(r)]$
73: ELSE $vval[a] \leftarrow vval[a] \bullet \langle p, V, UI_p\rangle$
74: $rnd[a] \leftarrow vrnd[a] \leftarrow r$
75: $UI_a \leftarrow USIG^A[a]$.createUI($\langle$"2b", $r, vval[a]\rangle$)
76: $\langle$"2b", $r, vval[a], UI_a\rangle \Rightarrow L$

77: $Learn(l) \triangleq$
78: **pre-conditions:**
79: $l \in learners$
80: $\exists Q \subseteq A$:
81: Q is a quorum
82: $\forall a \in Q, \langle$"2b", $r, \_, UI_a\rangle \Leftarrow a \wedge$
83: verifyUI(PK_a,UI_a,$\langle$"2b", $r, \_\rangle$) $\wedge$
84: verifyCnt(UI_a,$cnt[a]$)
85: **actions:**
86: LET $P \subset CF(r) : \forall p \in P, \langle$"2a", $r, \langle p, Nil\rangle, UI_p\rangle \Leftarrow p \wedge$ verifyUI(PK_p,UI_p,$\langle p, Nil\rangle$) $\wedge$ verifyCnt(UI_p,$cnt[p]$)
87: $Q2bVals = [v : \langle$"2b", $r, v, UI_a\rangle \Leftarrow a \in Q \wedge$
88: $\forall \langle q, W, UI_q\rangle \in v$: verifyUI($PK_q$,$UI_q$,$\langle p, W\rangle$) $\wedge$
89: verifyCnt(UI_q,$cnt[q]$]
90: $w = \sqcap Q2bVals \bullet [\langle u, Nil, UI_u\rangle : u \in P]$
91: $learned[l] = learned[l] \sqcup w$

ou não envia nada.

Sendo assim, cada *collision-fast proposer* pode fazer uma única proposta para uma determinada *rodada* r. Caso a proposta seja Nil, então essa pode ser enviada diretamente para os *learners*, caso contrário, deve ser enviada para os *acceptors* para que fique armazenada caso seja necessário um novo *round* para finalizar aquela instância do consenso.

Desta forma, o algoritmo necessita de apenas 2 passos de comunicação para um valor proposto ser decido, sendo rápido à despeito de colisões mesmo na presença de agentes maliciosos.

Além disso, apenas $2f + 1$ *acceptors* são necessários, pois os mesmos não conseguem modificar uma proposta feita por um *CF-proposer*, pois o identificador único gerado pelo *proposer* sempre acompanha a proposta. Tampouco conseguem enviar diferentes mensagens para diferentes *learners*.

6. Conclusão

De forma geral as principais contribuições feitas neste trabalho são três: i) proposta de uma versão modificada do protocolo *Collision Fast Atomic Broadcast* que tolera falhas bizantinas mas que não é rápido a despeito de falhas bizantinas; ii) conjectura da impossibilidade de um protocolo de Consenso Bizantino rápido à despeito de falhas bizantinas em sistema parcialmente síncrono; e, iii) proposta de um algoritmo rápido à despeito de falhas bizantinas, que contorna a impossibilidade apresentada com o uso do componente seguro USIG (*Unique Sequential Identifier Generator*).

Em trabalhos futuros pretendemos primeiramente provar formalmente a conjectura apresentada aqui. Em segundo lugar, pretendemos avaliar experimentalmente os protocolos propostos, comparando-os com outros na literatura.

Referências

Burrows, M. (2006). The chubby lock service for loosely-coupled distributed systems.

Camargos, L. J., Schmidt, R. M., and Pedone, F. (2007). Multicoordinated paxos. In *Proceedings of the Twenty-sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '07, pages 316–317, New York, NY, USA. ACM.

Castro, M. and Liskov, B. (2002). Practical Byzantine fault-tolerance and proactive recovery. *ACM Transactions on Computer Systems*, 20(4):398–461.

Chun, B.-G., Maniatis, P., Shenker, S., and Kubiatowicz, J. (2007). Attested append-only memory: Making adversaries stick to their word. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, SOSP '07, pages 189–204, New York, NY, USA. ACM.

Correia, M., Neves, N. F., and Verissimo, P. (2004). How to tolerate half less one byzantine nodes in practical distributed systems. In *Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems*, SRDS '04, pages 174–183, Washington, DC, USA. IEEE Computer Society.

Du, J., Sciascia, D., Elnikety, S., Zwaenepoel, W., and Pedone, F. (2014). Clock-RSM: Low-latency inter-datacenter state machine replication using loosely synchronized physical clocks. In *44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2014, Atlanta, GA, USA, June 23-26, 2014*, pages 343–354.

Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565.

Lamport, L. (1998). The part-time parliament. *ACM TRANSACTIONS ON COMPUTER SYSTEMS*, 16(2):133–169.

Lamport, L. (2006a). Fast paxos. *Distributed Computing*, 19(2):79–103.

Lamport, L. (2006b). Lower bounds for asynchronous consensus. *Distributed Computing*, 19(2):104–125.

Lampson, B. (1996). How to build a highly available system using consensus. In Babaoglu, O. and Marzullo, K., editors, *Distributed Algorithms*, volume 1151 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg.

Mao, Y., Junqueira, F. P., and Marzullo, K. (2008). Mencius: Building efficient replicated state machines for wans. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, OSDI'08, pages 369–384, Berkeley, CA, USA. USENIX Association.

Martin, J.-P. and Alvisi, L. (2006). Fast byzantine consensus. *IEEE Trans. Dependable Secur. Comput.*, 3(3):202–215.

Saramago, R. Q. (2016). Implementing and evaluating the collision-fast atomic broadcast. Master's thesis, Faculty of Computing, Federal University of Uberlândia, Brazil.

Schmidt, R., Camargos, L., and Pedone, F. (2014). Collision-fast atomic broadcast. In *Proceedings of the 2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, AINA '14, pages 1065–1072, Washington, DC, USA. IEEE Computer Society.

Veronese, G. S., Correia, M., Bessani, A. N., Lung, L. C., and Verissimo, P. (2013). Efficient byzantine fault-tolerance. *IEEE Trans. Comput.*, 62(1):16–30.

Uma Solução de Difusão Confiável Hierárquica em Sistemas Distribuídos Assíncronos

Luiz A. Rodrigues[1], Denis Jeanneau[2], Elias P. Duarte Jr.[3] e Luciana Arantes[2]

[1] Colegiado de Ciência da Computação – Unioeste-Cascavel, Brasil

[2]Sorbonne Universités, UPMC, CNRS, Inria, França

[3]Departamento de Informática – UFPR, Brasil

luiz.rodrigues@unioeste.br, denis.jeanneau@lip6.fr

Abstract. *Reliable broadcast is a key communication service for the development of distributed systems, since it allows messages to be sent simultaneously from one source to all other processes in the system. This paper presents a hierarchical reliable broadcast algorithm built over a virtual topology called VCube. Messages are propagated by spanning trees that self-adapt dynamically as soon as a crash is detected by an underlying monitoring system. Processes can fail by crashing and a failure is permanent. Special messages are used to treat false suspicions occasionally generated due to the asynchronous system model. Experimental results show the efficiency of the proposed solution compared to a one-to-all strategy.*

Resumo. *Difusão confiável é um serviço fundamental de comunicação para o desenvolvimento de sistemas distribuídos, pois permite o envio simultâneo de mensagens de um processo fonte para todos os demais processos do sistema. Este trabalho apresenta uma solução hierárquica para a difusão confiável de mensagens construída sobre a topologia virtual denominada VCube. As mensagens são propagadas por árvores geradoras que se adaptam dinamicamente à medida que falhas são detectadas pelo mecanismo de monitoramento. Os processos podem falhar por parada (crash) e uma falha é permanente. Mensagens especiais são utilizadas para tratar falsas suspeitas ocasionalmente geradas em virtude do modelo de sistema assíncrono. Resultados experimentais mostram a eficiência da solução proposta comparada com uma estratégia um-para-todos.*

1. Introdução

Difusão de mensagens (*broadcast*) é um componente básico para implementar diversos algoritmos e serviços distribuídos como notificação, entrega de conteúdo, replicação e comunicação em grupo [Leitão et al. 2007, Yang et al. 2009, Bonomi et al. 2013]. Um processo em um sistema distribuído utiliza difusão para enviar uma mensagem a todos os outros processos do sistema. No entanto, se este processo falha durante o procedimento de difusão, alguns processos podem receber a mensagem enquanto outros não. A difusão de melhor-esforço garante que, se o emissor é correto, todos os processos corretos recebem a mensagem difundida por ele. Por outro lado, se o emissor pode falhar, estratégias de difusão confiável (*reliable broadcast*) precisam ser implementadas [Hadzilacos e Toueg 1993].

Algoritmos de difusão tolerante a falhas são normalmente implementados utilizando enlaces ponto-a-ponto confiáveis e primitivas SEND e RECEIVE. Os processos invocam BROADCAST(m) e DELIVER(m) para difundir e entregar uma mensagem m para/de outros processos da aplicação, respectivamente. Para incluir tolerância a falhas, um detector de falhas [Freiling et al. 2011] pode ser utilizado para notificar o algoritmo de *broadcast*, que deve reagir apropriadamente quando uma falha é detectada.

Este trabalho apresenta um algoritmo de difusão confiável no qual cada processo do sistema é alcançado por meio de uma árvore dinâmica construída e mantida sobre uma topologia baseada em hipercubo virtual chamada VCube [Duarte et al. 2014]. O sistema é representado logicamente por um grafo completo com enlaces confiáveis. No VCube, os processos são organizados em *clusters* progressivamente maiores, formando um hipercubo completo quando não há processos falhos. Em caso de falhas, os processos corretos são reconectados entre si mantendo-se as propriedades logarítmicas do hipercubo. Uma versão preliminar do algoritmo foi apresentada em [Jeanneau et al. 2016]. Este artigo apresenta o algoritmo em detalhes e inclui resultados de simulação em cenários diversos.

Além da especificação, o algoritmo proposto foi comparado com uma estratégica um-para-todos ponto-a-ponto. Os resultados confirmam a eficiência da solução de broadcast proposta considerando: (1) a latência para entregar a mensagem a todos os processos corretos; e (2) o número total de mensagens, incluindo retransmissões em caso de falhas.

O restante do texto está organizado nas seguintes seções. A Seção 2 discute os trabalhos correlatos. A Seção 3 apresenta as definições básicas, o modelo do sistema e o VCube. O algoritmo de *broadcast* confiável proposto é apresentado na Seção 4. Os resultados de simulação são apresentados na Seção 5. A Seção 6 apresenta a conclusão e os trabalhos futuros.

2. Trabalhos Relacionados

Grande parte dos algoritmos de difusão são baseados em árvores geradoras. Schneider et al. (1984) introduziram um algoritmo de difusão tolerante a falhas no qual a raiz é o processo que inicia a transmissão, ou seja, o remetente. Se um processo p que pertence à árvore falhar, outro processo assume a responsabilidade de retransmitir as mensagens que p deveria ter transmitido se estivesse correto. Os processos podem falhar por *crash* e a falha de um processo é detectada por um módulo de detecção de falhas após um intervalo de tempo finito, mas não conhecido. Um processo pode enviar uma próxima mensagem somente após a difusão anterior ter sido concluída. No entanto, os autores não descrevem como a detecção de falhas é implementada, tampouco fornecem um algoritmo para construir e reorganizar a árvore após a falha.

Garcia-Molina e Kogan (1988) implementaram a difusão confiável utilizando o mecanismo de *multicast* não-confiável em uma rede assíncrona ponto-a-ponto. Listas de prioridade são usadas para especificar a ordem em que nodos devem acessar a rede depois de uma falha e as listas são baseadas em informações sobre a topologia da rede.

Ramanathan e Shin (1988) propuseram uma difusão confiável que é executada em um hipercubo e usa árvores geradoras disjuntas para o envio de uma mensagem através de vários caminhos. Algoritmos de caminhos múltiplos são particularmente úteis em sistemas que não podem tolerar a sobrecarga de tempo para a detecção de processos com falhas, mas há uma sobrecarga no número de mensagens duplicadas.

Em Wu (1996), os autores apresentam um algoritmo de difusão tolerante a falhas para hipercubos baseado em árvores binomiais. O algoritmo pode recursivamente regenerar uma subárvore falha, induzida por um nodo falho, através de uma das folhas da árvore. No entanto, ao contrário da abordagem proposta neste trabalho, a solução exige uma mensagem especial para indicar que a árvore deverá ser reorganizada e, neste caso, as mensagens de difusão não são tratadas pelos nodos até que a árvore seja reconstruída.

Liebeherr e Beam (1999) apresentam um procolo, chamado HyperCast, que organiza os membros de um grupo *multicast* em um hipercubo lógico usando o código *gray* para ordená-los. A árvore é sobreposta no hipercubo para evitar implosão de ACKs. O processo com o identificador mais alto é considerado a raiz da árvore. Entretanto, em função de falhas, múltiplos nodos podem considerar a si próprios como raiz e/ou diferentes nodos podem ter visões diferentes sobre a identidade da raiz.

Um protocolo híbrido combinando estratégias de árvore e *gossip* foi proposto por Leitão et al. (2007). Nesta solução, chamada HyParView, uma árvore de difusão é criada sobre uma rede *gossip*. Outras soluções utilizam *gossiping* para criar protocolos probabilísticos. Eugster et al. (2003) propuseram um algoritmo no qual cada processo conhece um número fixo de vizinhos escolhidos aleatoriamente. Pereira et al. (2004) propuseram um protocolo epidêmico no qual as mensagens são enviadas usando os nodos com maior capacidade de transmissão, em uma tentativa de otimizar a taxa de disseminação.

Em Rodrigues et al. (2014) foi apresentada uma solução para *broadcast* confiável utilizando árvores dinâmicas no VCube. O algoritmo permite a propagação de mensagens utilizando múltiplas árvores construídas dinamicamente a partir de cada emissor e que incluem todos os nodos do sistema. Diferente da solução proposta neste trabalho, o modelo é síncrono e o detector de falhas é perfeito.

3. Definições e Modelo do Sistema

Um sistema distribuído é composto por um conjunto finito P com $n > 1$ processos $\{p_0, .., p_{n-1}\}$ que se comunicam por troca de mensagens. Considera-se que cada processo executa uma tarefa e está alocado em um nodo distinto. Assim, os termos *nodo* e *processo* são utilizados com o mesmo sentido.

Comunicação. Os processos se comunicam através do envio e recebimento de mensagens. A rede é representada por um grafo completo, isto é, cada par de processos está conectado diretamente por enlaces ponto-a-ponto bidirecionais. No entanto, processos são organizados em uma topologia de hipercubo virtual, chamada VCube. Se não existem processos falhos, VCube é um hipercubo completo. Após uma falha, a topologia do VCube é modificada dinamicamente (mais detalhes na Seção 3.1). As operações de envio e recebimento são atômicas. Enlaces são confiáveis, garantindo que as mensagens trocadas entre dois processos nunca são perdidas, corrompidas ou duplicadas pelos enlaces.

Modelo de Falhas. O sistema admite falhas do tipo *crash* permanente. Um processo que nunca falha é considerado *correto* ou *sem-falha*. Caso contrário ele é considerado *falho*.

Detecção de falhas. O sistema é assíncrono, isto é, não existem limites conhecidos para a velocidade de processamento e atraso de transmissão de mensagens. A propriedade de completude do detector implementado pelo VCube é garantida, mas a acurácia não. Ou seja, todos os processos corretos detectam em algum momento a falha de um processo, mas falsas suspeitas podem acontecer arbitrariamente [Freiling et al. 2011].

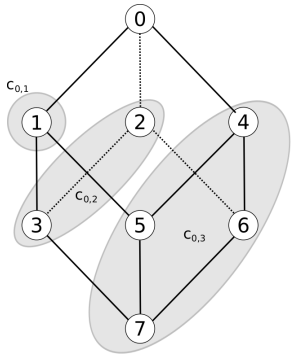
3.1. O VCube

O VCube é uma topologia baseada em hipercubo virtual criada e mantida com base nas informações de diagnóstico obtidas por meio de um sistema de monitoramento de processos descrito em Duarte Jr. et al. (2014). Cada processo que executa o VCube é capaz de testar outros processos no sistema para verificar se estão corretos ou falhos. Um processo é considerado *correto* ou *sem-falha* se a resposta ao teste for recebida corretamente dentro do intervalo de tempo esperado. Caso contrário, o processo é considerado *falho* ou suspeito. Os processos são organizados em *clusters* progressivamente maiores. Cada *cluster* $s = 1, .., \log_2 n$ possui 2^s elementos, sendo n o total de processos no sistema. Os testes são executados em rodadas. Para cada rodada um processo i testa o primeiro processo sem-falha j na lista de processos de cada *cluster* s e obtém dele as informações que ele possui sobre os demais processos do sistema.

Os membros de cada *cluster* s e a ordem na qual eles são testados por um processo i são obtidos da lista gerada pela função $c_{i,s}$, definida a seguir. O símbolo $\oplus$ representa a operação binária de OU exclusivo (XOR):

$$c_{i,s} = (i \oplus 2^{s-1}, c_{i\oplus 2^{s-1},1}, ..., c_{i\oplus 2^{s-1},s-1}) \quad (1)$$

A Figura 1 exemplifica a organização hierárquica dos processos em um hipercubo de três dimensões com $n = 2^3$ elementos. A tabela da direita apresenta os elementos de cada *cluster* $c_{i,s}$. Como exemplo, na primeira rodada o processo p_0 testa o primeiro processo no *cluster* $c_{0,1} = (1)$ e obtém informações sobre o estado dos demais processos armazenada em p_1. Em seguida, p_0 testa o processo p_2, que é primeiro processo no *cluster* $c_{0,2} = (2, 3)$. Por fim, p_0 executa testes no processo p_4 do *cluster* $c_{0,3} = (4, 5, 6, 7)$. Como cada processo executa estes procedimentos de forma concorrente, ao final da última rodada todo processo será testado ao menos uma vez por um outro processo. Isto garante que em $\log_2^2 n$ rodadas, todos os processos terão localmente a informação atualizada sobre o estado dos demais processos no sistema (latência de diagnóstico).



s	$c_{0,s}$	$c_{1,s}$	$c_{2,s}$	$c_{3,s}$	$c_{4,s}$	$c_{5,s}$	$c_{6,s}$	$c_{7,s}$
1	1	0	3	2	5	4	7	6
2	2 3	3 2	0 1	1 0	6 7	7 6	4 5	5 4
3	4 5 6 7	5 4 7 6	6 7 4 5	7 6 5 4	0 1 2 3	1 0 3 2	2 3 0 1	3 2 1 0

Figura 1. Organização Hierárquica do VCube de $d = 3$ dimensões com os *clusters* do processo 0 (esq.) e a tabela completa da $c_{i,s}$ (dir.)

4. O Algoritmo de Difusão Confiável Proposto

Um algoritmo de *broadcast* confiável garante que uma mensagem enviada por um processo emissor (fonte) é entregue (*delivered*) a todos os processos corretos, mesmo se o emissor falhar durante o procedimento de difusão. Para tanto, três propriedades devem ser satisfeitas [Kshemkalyani e Singhal 2008]:

- **Entrega confiável** (*validity*): se um processo correto i envia uma mensagem m, ele também entrega m em um tempo finito;
- **Integridade** (*integrity*): toda mensagem m é entregue por todos os processos corretos no máximo uma vez (não-duplicação) e somente se m foi previamente enviada por algum outro processo (não-criação);
- **Acordo** (*agreement*): se um processo correto entregou a mensagem m, então todo processo correto fará a entrega de m em um tempo finito.

4.1. Funções, Mensagens e Variáveis Locais

Com base na organização lógica do VCube e na função $c_{i,s}$ foram definidas as seguintes funções. Seja i um processo que executa o algoritmo de *broadcast* e $d = \log_2 n$ a dimensão do d-VCube com 2^d processos. A lista de processos considerados corretos por i é armazenada em $correct_i$.

A função $cluster_i(j) = s$ calcula o identificador s do *cluster* do processo i que contém o processo j, $1 \leq s \leq d$. Por exemplo, considerando o 3-VCube da Figura 1, $cluster_0(1) = 1$, $cluster_0(2) = cluster_0(3) = 2$ e $cluster_0(4) = cluster_0(5) = cluster_0(6) = cluster_0(7) = 3$.

Seja m a mensagem de aplicação a ser transmitida de um processo emissor (fonte), para todos os outros processos no sistema. Três tipos de mensagens são utilizadas:

- $\langle TREE, m \rangle$: a mensagem de *broadcast* que carrega a mensagem de aplicação e que deve ser retransmitida sobre a árvore gerada com base na topologia do VCube;
- $\langle DELV, m \rangle$: mensagem enviada ao processo suspeito para contornar falsas suspeitas. Em caso de falsa suspeita, essa mensagem é processada pelo receptor considerado falho, mas não é retransmitida na árvore;
- $\langle ACK, m \rangle$: mensagens utilizadas para confirmar o recebimento das mensagens $\langle TREE, m \rangle$.

Para simplificar, as mensagens serão identificadas por `TREE`, `DELV` e `ACK`.

Cada mensagem m contém ainda dois parâmetros: (1) o identificador da origem, isto é, o processo que iniciou a difusão, obtido com a função $source(m)$; e (2) o *timestamp*, um contador sequencial local que identifica de forma única cada mensagem gerada em um processo, obtido pela função $ts(m)$.

As variáveis locais mantidas pelos processos são:

- $correct_i$: conjunto dos processos considerados corretos pelo processo i;
- $last_i[n]$: a última mensagem recebida de cada processo fonte. $last_i[j]$ é a última mensagens difundida por j que foi entregue por i;
- ack_set_i: o conjunto com todos os ACKs pendentes no processo i. Para cada mensagem $\langle TREE, m \rangle$ recebida pelo processo i de um processo j e retransmitida para o processo k, um elemento $\langle j, k, m \rangle$ é adicionado a este conjunto. O símbolo $\perp$ representa um elemento nulo. O asterisco é usado como curinga para selecionar ACKs no conjunto ack_set. Um elemento $\langle j, *, m \rangle$, por exemplo, representa todos os ACKs pendentes para uma mensagem m recebida pelo processo j e retransmitida para qualquer outro processo.
- $pending_i$: lista de mensagens m recebidas pelo processo i de um processo fonte $source(m)$ que ainda não podem ser entregues à aplicação por estarem fora de ordem, isto é, $ts(m) > ts(last_i(source(m)) + 1$.

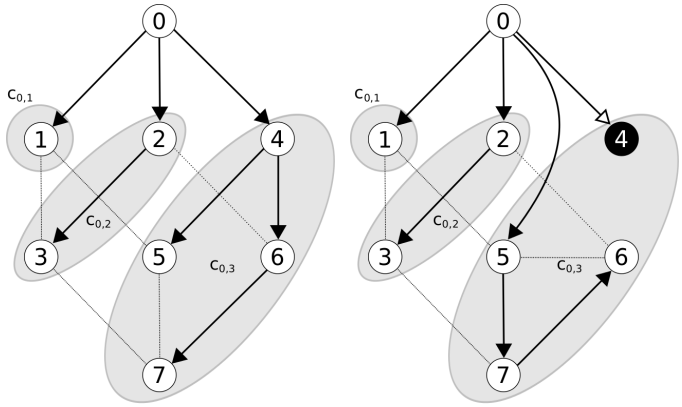
- $history_i$: histórico de mensagens que já foram retransmitidas por i. Este conjunto é utilizado para prevenir o envio duplicado de mensagens no mesmo *cluster*. $\langle j, m, h\rangle \in history_i$ indica que a mensagem m recebida do processo j já foi enviada por i para o *cluster* $c_{i,s}$ para todo $s \in [1, h]$.

4.2. Descrição do Algoritmo

O Algoritmo 1 apresenta uma solução para difusão confiável baseada em árvores geradores construídas dinamicamente com base no grafo de testes do VCube.

Considere a execução do sistema em um cenário sem processos falhos. Um processo i inicia o *broadcast* invocando a função BROADCAST(m). A linha 7 garante que um novo *broadcast* será iniciado apenas após o término do anterior, isto é, quando não há mais *acks* pendentes para a mensagem $last_i[i]$. Quando a mensagem m é entregue localmente em i (linha 9) e, pela invocação de BROADCAST_TREE (linha 10), i retransmite m para os vizinhos no VCube. Para isso, ele invoca a função BROADCAST_CLUSTER para cada *cluster* $s \in [1, log_2 n]$, que envia `TREE` para o primeiro processo k considerado correto no *cluster* (linha 27). Para cada mensagem `TREE` enviada, um *ack* é incluído na lista de *acks* pendentes (linha 28). A medida que os *acks* retornam, as pendências são removidas e os `ACK`s são propagados até a raiz.

A Figura 2(a) mostra uma execução sem falhas considerando o processo 0 (p_0) como fonte. Após fazer a entrega local, p_0 envia uma cópia da mensagem para p_1, p_2 e p_4, que são os vizinhos dele no VCube. Estes, por sua vez, retransmitem a mensagem na subárvore formada pelos *clusters* internos, isto é, aqueles contidos dentro do *cluster* ao qual estão em relação ao processo fonte i. No exemplo, p_2 retransmite para p_3; p_4 retransmite para p_5 e p_6; e p_6 retransmite para p_7.



(a) Sem falhas (b) p_4 é suspeito

Figura 2. Difusão confiável no processo 0 (p_0).

Considere agora a execução do algoritmo em um cenário com processos falhos ou suspeitos. Considere ainda que uma difusão está em andamento, mas que o processo j ainda não foi suspeito. Se o emissor i está diretamente conectado com j, irá enviar a mensagem `TREE` para j e aguardar o `ACK`. Se j está realmente falho, a sua falha será detectada em algum momento pelo VCube executando em i, que fará a notificação. Quando j é detectado como falho pelo processo i o evento de CRASH(j) é automaticamente executado. Imediatamente, todos as mensagens enviadas a j e que ainda não foram confirmadas

Algoritmo 1 Algoritmo de Difusão Confiável no Processo i

```
1:  last_i[n] ← {⊥, ..., ⊥}
2:  ack_set_i ← ∅
3:  correct_i ← {0, ..., n − 1}
4:  pending_i ← ∅
5:  history_i ← ∅

6:  procedure BROADCAST(msg m)
7:      wait until ack_set_i ∩ {⟨⊥, ∗, last_i[i]⟩} = ∅
8:      last_i[i] ← m
9:      DELIVER(m)
10:     BROADCAST_TREE(⊥, m, log_2 n)

11: procedure BROADCAST_TREE(proc j, msg m, int h)
12:     start ← 0
13:     if ∃x : ⟨j, m, x⟩ ∈ history_i then
14:         start ← x
15:         history_i ← history_i \ {⟨j, m, x⟩}
16:     history_i ← history_i ∪ {⟨j, m, max(start, h)⟩}
17:     if start < h then
18:         for all s ∈ [start + 1, h] do
19:             BROADCAST_CLUSTER(j, m, s)

20: procedure BROADCAST_CLUSTER(proc j, msg m, int s)
21:     sent ← false
22:     for all k ∈ c_{i,s} do
23:         if sent = false then
24:             if ⟨j, k, m⟩ ∈ ack_set_i
                  and k ∈ correct_i then
25:                 sent ← true
26:             else if k ∈ correct_i then
27:                 SEND(⟨TREE, m⟩) to p_k
28:                 ack_set_i ← ack_set_i ∪ {⟨j, k, m⟩}
29:                 sent ← true
30:             else if ⟨j, k, m⟩ ∉ ack_set_i then
31:                 SEND(⟨DELV, m⟩) to p_k

32: procedure CHECK_ACKS(proc j, msg m)
33:     if j ≠ ⊥ and ack_set_i ∩ {⟨j, ∗, m⟩} = ∅ then
34:         SEND(⟨ACK, m⟩) to p_j

35: procedure HANDLE_MESSAGE(proc j, msg m)
36:     pending_i ← pending_i ∪ {m}
37:     while ∃l ∈ pending_i : source(l) = source(m)
          ∧(ts(l) = ts(last_i[source(l)]) + 1
          or last_i[source(l)] = ⊥ ∧ ts(l) = 0) do
38:         last_i[source(l)] ← l
39:         pending_i ← pending_i \ {l}
40:         DELIVER(l)
41:     if source(m) ∉ correct_i then
42:         BROADCAST_TREE(j, last_i[source(m)], log_2 n)

43: upon RECEIVE ⟨TREE, m⟩ from p_j
44:     HANDLE_MESSAGE(m)
45:     BROADCAST_TREE(j, m, cluster_i(j) − 1)
46:     CHECK_ACKS(j, m)

47: upon RECEIVE ⟨DELV, m⟩ from p_j
48:     HANDLE_MESSAGE(m)

49: upon RECEIVE ⟨ACK, m⟩ from p_j
50:     for all k = x : ⟨x, j, m⟩ ∈ ack_set_i do
51:         ack_set_i ← ack_set_i \ {⟨k, j, m⟩}
52:         CHECK_ACKS(k, m)

53: upon notifying CRASH(j)
54:     correct_i ← correct_i \ {j}
55:     for all p = x, m = y :
          ⟨x, j, y⟩ ∈ ack_set_i ∩ {⟨∗, j, ∗⟩} do
56:         BROADCAST_CLUSTER(p, m, cluster_i(j))
57:         ack_set_i ← ack_set_i \ {⟨p, j, m⟩}
58:         CHECK_ACKS(p, m)
59:     if last_i[j] ≠ ⊥ then
60:         BROADCAST_TREE(j, last_i[j], log_2 n)

61: upon notifying UP(j)
62:     correct_i ← correct_i ∪ {j}
```

(aquelas registradas em ack_set_i) são retransmitidas ao próximo processo considerado correto no mesmo *cluster* de i, se existir um. Além disso, o $broadcast$ da última mensagem recebida j é refeito utilizando a raiz de i (linha 60). Isso garante que que todos os processos corretos receberão uma cópia da última mensagem de j, mesmo que o *broadcast* de j tenha sido interrompido antes do término. Note que isso é feito por todos os processos corretos a medida que a notificação da falha de j é detectada, pois não há como determinar qual processo possui a mensagem mais atual de j.

Em um segundo cenário de falhas, considere que o processo j já havia sido suspeito antes do processamento da mensagem `TREE`. No caso do emissor, uma mensagem `TREE` será enviada para o primeiro processo correto k de cada *cluster* s conforme a lista ordenada gerada pela $c_{i,s}$. Para os processos considerados falhos posicionados antes de k na lista, uma mensagem `DELV` será enviada. Em caso de falsa suspeita, `DELV` é processada, mas não é retransmitida na subárvore de j. No exemplo da Figura 2(b) p_4 está suspeito por p_0. Sendo $c_{0,3} = (4, 5, 6, 7)$, o processo fonte p_0 envia `TREE` para p_5, que é o primeiro sem falha na lista, e `DELV` para p_4. A difusão continua na subárvore de p_5 que entrega a mensagem a todos os corretos no *cluster*.

4.3. Prova de Correção

O correto funcionamento do Algoritmo 1 como uma solução de difusão confiável é garantido pelas propriedades de entrega confiável, integridade (não-duplicação e não-criação) e acordo.

Lema 1 (Entrega Confiável)**.** *O Algoritmo 1 garante que se um processo fonte correto i envia uma mensagem m por difusão, ele também entrega m em um tempo finito.*

Prova. Se um processo i realiza o *broadcast* de uma mensagem m, a única maneira de i não entregar m é se i aguardar indefinidamente na linha 7. Esta espera é interrompida quando o conjunto ack_set_i não contém mais mensagens de confirmação pendentes em relação a mensagem $last_i[i]$ difundida previamente por i.

Para todo processo j que i envia $last_i[i]$, i adiciona um *ack* pendente em ack_set_i (linha 28). Se j está correto, ele responde em um tempo finito com uma mensagem ACK (linha 34) e i remove $\langle \perp, j, last_i[j] \rangle$ de ack_set_i na linha 51. Se j está falho, em um tempo finito i será notificado da falha e removerá o *ack* pendente na linha 57.

Como resultado, todos os *acks* pendentes para $last_i[i]$ serão removidos do conjunto ack_set_i em um tempo finito e i realizará a entrega de m na linha 9.

Lema 2. *Para quaisquer processos i e j, o valor de $ts(last_i[j])$ é sempre incrementado ao longo do tempo.*

Prova. Por questões de simplificação, considere $ts(\perp) = -1$. O vetor $last_i$ é modificado somente nas linhas 8 e 38.

A modificação na linha 8 é feita somente quando i inicia o *broadcast* de uma nova mensagem m. Como o *timestamp* de uma nova mensagem enviada por um mesmo processo é sempre incrementado, $ts(m) > ts(last_i[i])$. Quando i invoca BROADCAST com m, $ts(last_i[i])$ será incrementado na linha 8.

Na segunda possibilidade, $last_i$ é modificado na linha 38. $last_i[source(l)]$ é atualizado com a mensagem l se $last_i[source(l)] = \perp$ e $ts(l) = 0$ (e portanto, $ts(last_i[source(l)]) = -1 < ts(l)$), ou se $ts(l) = ts(last_i[source(l)]) + 1$. Neste caso, $last_i[source(l)$ é atualizado somente se o novo valor de $ts(last_i[source(l)])$ é maior que o atual.

Lema 3 (Integridade)**.** *O Algoritmo 1 garante que para toda mensagem m é entregue por todos os processos corretos no máximo uma vez (não-duplicação) e somente se m foi previamente enviada por algum outro processo (não-criação).*

Prova. Os processos somente entregam uma mensagem se eles estão iniciando o *broadcast* (linha 9) ou se a mensagem está no conjunto $pending_i$ (linha 40). Mensagens são adicionadas no conjunto $pending_i$ somente na linha 36, depois de serem recebidas de outro processo. Considerando que os enlaces são confiáveis e que não criam mensagens, uma mensagem é recebida somente se foi previamente difundida.

Para mostrar que não há entrega duplicada, considere os seguintes casos:

- $\mathbf{source(m) = i}$. O processo i invocou BROADCAST com parâmetro m. Como provado pelo Lema 1, i entregará m na linha 9. Uma vez que BROADCAST é invocado uma única vez para cada mensagem, a única forma de i entregar m uma segunda vez é na linha 40. Como $last_i[i]$ foi atualizada com m na linha 8, o Lema 2 garante que m nunca satisfará a condição da linha 37.

- $\mathbf{source(m) \neq i}$. O processo i não é o emissor (fonte) da mensagem m e, portanto, não invocou BROADCAST(m). Neste caso, a única forma para i entregar m é na linha 40. Antes de i fazer a entrega de m pela primeira fez, ele atualiza o conjunto $last_i[source(m)]$ com m na linha 38. Novamente, a partir do Lema 2, a mensagem m jamais satisfará a condição de teste da linha 37 e, portanto, i poderá entregá-la uma única vez.

Lema 4 (Acordo)**.** *O Algoritmo 1 garante que se um processo correto entregou a mensagem m, então todo processo correto fará a entrega de m em um tempo finito.*

Prova. Seja m a mensagem de *broadcast* enviada por um processo i. Duas situações devem ser consideradas:

- **i é correto**. A prova por indução mostrará que cada processo correto recebe m.
  Como base da indução, considere um sistema com $n = 2$ processos e $P = \{i, j\}$. Neste caso, $c_{i,1} = \{j\}$. Portanto, i enviará a mensagem m para j na linha 31 se i suspeita j ou na linha 27 caso contrário. Se j está correto, ele receberá a mensagem m em um tempo finito (uma vez que os canais são confiáveis) e entregará m na linha 40. i também fará a entrega de m, por conta da propriedade de entrega confiável.
  Considere como hipótese que todo processo correto recebe m para um sistema com $n = 2^k$ processos. O passo da indução provará que a hipótese é válida para $n = 2^{k+1}$. O sistema com 2^{k+1} pode ser visto como dois subsistemas $P_1 = \{i\} \cup \bigcup_{x=1}^{k} c_{i,x}$ e $P_2 = c_{i,k+1}$ tal que $|P_1| = |P_2| = 2^k$.
  Os procedimentos BROADCAST_TREE e BROADCAST_CLUSTER garantem que para cada cluster $s \in [1, k+1]$, i enviará m para ao menos um processo em $c_{i,s}$. Seja j o primeiro processo na $c_{i,k+1}$. Se j está correto, ele receberá m em um tempo finito. Se j está falho e i já foi notificado, i enviará a mensagem de qualquer forma para os casos de falsas suspeitas (linha 31), mas a enviará também para o próximo processo na $c_{i,k+1}$ (linha 27), que, estando correto, terá a tarefa de propagar a mensagem na subárvore daquele *cluster*. i repetirá esse procedimento até enviar `TREE` para um processo não suspeito na $c_{i,k+1}$, ou até enviar `DELV` para todos os processos na $c_{i,k+1}$.
  Se j está falho e i for notificado somente após ter enviado a mensagem, o procedimento BROADCAST_CLUSTER será invocado novamente na linha 56 assim que a falha for notificada, o que garante que i enviará a mensagem para um processo não-suspeito na $c_{i,k+1}$, se existir um. Como resultado, exceto se todos os processos estejam suspeitos, ao menos um processo correto receberá m. Este processo fará então a retransmissão de m na subárvore de P_2, conforme a linha 45.
  Uma vez que um processo correto retransmite a mensagem m nos dois subsistemas P_1 e P_2, e como cada subsistema possui 2^k processos, cada processo correto em P receberá a mensagem m em um tempo finito.

- **i está falho**. Se i falha antes de enviar m para algum outro processo correto, então nenhum processo entrega m e a propriedade de acordo está garantida. Se i falha após ter enviado a mensagem a todos os vizinhos corretos em cada *cluster*, a mensagem é entregue a todos os processos corretos em cada subárvore. Por outro lado, se i falha após enviar a mensagem m para alguns processos e um processo correto j recebe m, então j será notificado pelo detector de falhas em um tempo finito. Se j detectar a falha de i antes de receber a mensagem m, quando ele

recebê-la fará um novo *broadcast*, conforme a linha 42. Se j detectar a falha de i após o recebimento de m, o mesmo iniciará um *broadcast* completo na linha 60. Se j é correto, todo processo correto receberá m em um tempo finito.

Teorema 1. *O Algoritmo 1 é uma solução para difusão confiável. Ele garante as propriedades de entrega confiável, integridade e acordo.*

Prova. As propriedades de entrega confiável, integridade e acordo são garantidas pelo Lema 1, Lema 3 e Lema 4, respectivamente.

5. Avaliação Experimental

Nesta seção são apresentados os resultados dos experimentos de simulação realizados com o algoritmo de *broadcast* proposto. Os testes estão divididos em duas partes. Primeiro são apresentados os resultados para cenários sem processos falhos e, em seguida, para os cenários com falhas.

Para comparar a solução proposta, denominada VCUBE-RB, foi implementado uma solução um-para-todos, chamada ALL-RB, na qual o processo emissor (fonte) envia uma cópia da mensagem diretamente a cada processo do grupo e aguarda pelas confirmações (`ACKs`).

Os algoritmos foram implementados utilizando o Neko [Urbán et al. 2002], um *framework* Java para simulação de algoritmos distribuídos.

5.1. Parâmetros de Simulação

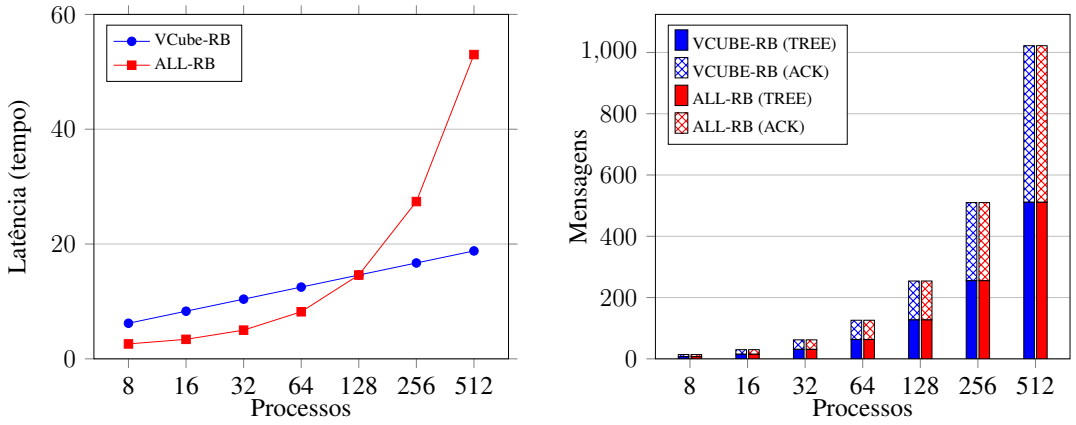
Quando um processo precisa enviar uma mensagem para mais de um destinatário ele deve utilizar primitivas SEND sequencialmente. Assim, para cada mensagem, t_s unidades de tempo são utilizadas para enviar a mensagem e t_r unidades para recebê-la, além do atraso de transmissão t_t. Estes intervalos são computados para cada cópia da mensagem enviada.

Para avaliar o desempenho de soluções de difusão, duas métricas foram utilizadas: (1) a latência para entregar a mensagem de *broadcast* a todos os processos corretos; e (2) total de mensagens enviadas pelo algoritmo, que incluem as mensagens `TREE`, `ACK` e `DELV`.

Os algoritmos propostos foram avaliados em diferentes cenários variando o número de processos e a quantidade de processos falhos. Os parâmetros de comunicação foram definidos em $t_s = t_r = 0.1$ e $t_t = 0.8$, sendo t_s o tempo de processamento no envio, t_t o tempo de transmissão e t_r o tempo de processamento no recebimento de cada mensagem. O intervalo de testes do detector foi definido em 30.0 unidades de tempo. Um processo é considerado falho se não responder ao teste após $4 * (t_s + t_r + t_t)$ unidades de tempo, isto é, 4.0.

5.2. Cenários sem Falhas

Por questões de simplificação, mas sem perda de generalidade, uma única mensagem de *broadcast* é enviada pelo processo 0 (p_0). A Figura 3 apresenta os resultados obtidos para sistemas com diferentes tamanhos. Na estratégia ALL-RB, a latência é menor para sistemas até 128 processos, mas aumenta rapidamente após este limiar em razão do atraso de processamento para o envio das cópias da mensagem `TREE` para cada um dos $n - 1$ processos corretos no sistema (lembre-se que cada mensagem enviada consome $t_s = 0.1$



(a) Latência (b) Total de Mensagens

Figura 3. Latência e total de mensagens em uma execução sem falhas.

unidades de tempo). Por outro lado, a raiz no VCUBE-RB envia $\log_2 n$ mensagens, uma para cada vizinho, sendo estas propagadas em paralelo nos demais níveis da árvore.

O total de mensagens é equivalente para VCUBE-RB e ALL-RB, conforme ilustrado na Figura 3(b), sendo igual ao dobro do número de processos no sistema menos um, isto é, $n - 1$ mensagens de aplicação (`TREE`) e $n - 1$ mensagens de confirmação (`ACK`). Como não há falhas e não foram geradas falsas suspeitas, nenhuma mensagem `DELV` é enviada pelo VCUBE-RB.

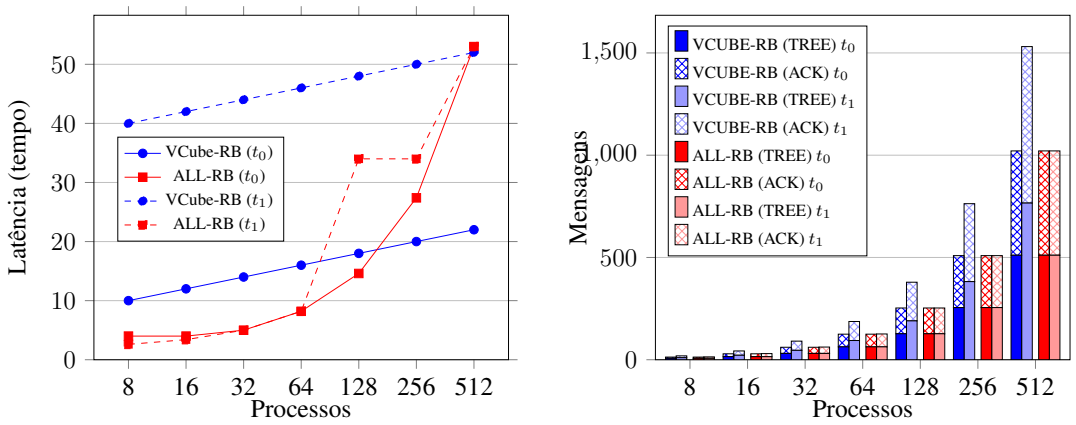
5.3. Cenários com Falhas

Os cenários com falhas foram divididos em dois casos. O primeiro considera a falha de um processo intermediário na árvore do VCUBE-RB e o segundo considera a falha do processo emissor. Por questões de simplificação, novamente um único processo emissor (p_0) realiza o *broadcast*.

Falha do Processo na Subárvore. O processo falho no nível intermediário da árvore é identificado por $p_{n/2}$. Este processo é o primeiro que recebe a mensagem no *cluster* de maior tamanho do sistema, o que pode acarretar na retransmissão de grande número de mensagens na subárvore do *cluster*. Em um VCube de 8 processos por exemplo, a falha do processo p_4, pode gerar retransmissões para p_5, p_6 e p_7, dependendo do momento em que a falha acontece. Para ALL-RB o mesmo processo foi utilizado na falha, embora o cenário seja equivalente para qualquer processo falho.

As falhas no processo $p_{n/2}$ foram simuladas em dois momentos distintos. No primeiro, a falha acontece no tempo $t_0 = 0.0$. Neste caso, nenhuma mensagem extra é gerada, visto que o processo falho não recebe a mensagem e, após a detecção da falha, uma nova mensagem será retransmitida para o próximo processo correto no *cluster*, que fará a retransmissão na subárvore de difusão uma única vez. No segundo experimento, a falha é configurada no tempo $t_1 = \log_2 n$. Este tempo é suficiente para que o processo intermediário receba e retransmita a mensagem `TREE` antes de falhar. Assim, quando a falha for detectada, uma nova mensagem será enviada para o próximo processo sem falha no *cluster* que fará uma nova difusão na subárvore daquele *cluster*. Cada processo correto no *cluster* do processo falho receberá a mensagem duas vezes. Note que a entrega é feita uma única vez em função dos *timestamps* das mensagens duplicadas.

A Figura 4 compara a latência e o número de mensagens geradas nos dois casos da falha do processo intermediário da árvore. A latência de difusão é aumentada em relação ao cenário sem falhas devido à latência de detecção do detector de falhas. Lembre-se que o *timeout* foi configurado em 4.0 unidades de tempo. No caso de ALL-RB, após a detecção da falha pelo processo fonte, a difusão é imediatamente concluída. No caso de VCUBE-RB, uma vez detectada a falha, a árvore é imediatamente reconfigurada e a difusão segue na subárvore do *cluster* do processo que falhou. Nestes cenários, além de possíveis mensagens adicionais, tem-se uma maior latência no VCUBE-RB nos sistemas menores, ainda próximos a 128. No entanto, a medida que o número de processos aumenta, a latência de ALL-RB é novamente comprometida pelo envio das múltiplas cópias da mensagem a partir do processo fonte.



(a) Latência (b) Total de Mensagens

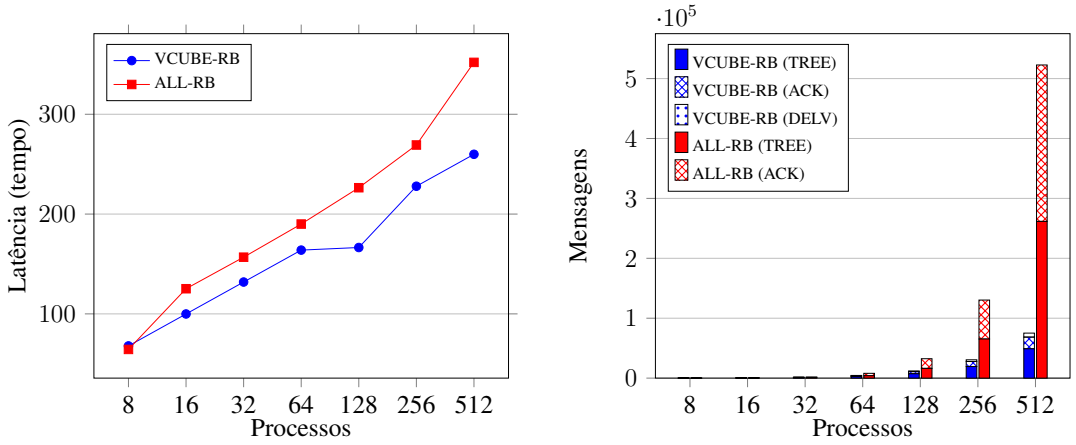
Figura 4. Latência e total de mensagens em uma execução com falha do nodo intermediário $p_{n/2}$ nos tempos $t_0 = 0.0$ e $t_1 = \log_2 n$.

Falha do Processo Emissor (Fonte). Considerando o pior caso, os cenários com falha foram comparados considerando apenas a falha do processo emissor (fonte), isto é, aquele que inicia o *broadcast* da mensagem. Para cada solução comparada, o processo zero (p_0) inicia a difusão de uma mensagem e falha logo após ter enviado uma cópia da mensagem para cada vizinho. Como cada envio consome $t_s = 0.1$, no ALL-RB a falha ocorre após o tempo $n * 0.1$ e para VCUBE-RB após $\log_2 n * 0.1$. Isso garante que todos os processos receberão uma cópia da mensagem e, portanto, terão que reiniciar o *broadcast* desta última mensagem recebida após a detecção da falha. Embora aconteçam em tempos diferentes para cada algoritmo, a falha do emissor é gerada na primeira rodada do VCube, não interferindo no tempo de detecção.

A Figura 5 apresenta os resultados obtidos considerando diferentes tamanhos de sistema. A latência difere pelo intervalo de tempo necessário para que cada processo propague a mensagem aos demais membros após serem notificados pelo VCube. Para a estratégia ALL-RB cada processo envia uma nova cópia da última mensagem de p_0 diretamente para todos os outros membros do sistema e aguarda pelas confirmações. Para a solução VCUBE-RB, cópias da última mensagem também são enviadas por cada processo do sistema, porém utilizando a árvore com raiz em cada processo.

A Figura 5(a) compara a latência das duas soluções implementadas. Embora o valor da latência seja próximo para ambos, VCUBE-RB apresenta menor latência a me-

dida que o número de processos cresce. Em relação ao total de mensagens, percebe-se na Figura 5(b) que o VCUBE-RB utilizou um número muito menor de mensagens. Este resultado é fruto do mecanismo de retransmissão na árvore, que evita que duas mensagens iguais sejam propagadas na mesma subárvore se a mesma já foi encaminhada e o ACK está pendente. Nota-se no gráfico um desequilíbrio entre o número mensagens TREE em relação as respectivas mensagens de confirmação ACK. Isto acontece em cenários com falhas porque quando um processo recebe uma mensagem TREE de um processo falho (fonte ou intermediário) ele não devolve o ACK. No caso da estratégia VCUBE-RB, por exemplo, a propagação do ACK até o processo emissor que está falho é interrompida assim que a falha é detectada por um processo no caminho reverso da árvore. Além disso, o VCUBE-RB não retransmite mensagens que já foram propagadas na árvore. Isso reduz consideravelmente o total de mensagens retransmitidas após a falha em comparação com a estratégia um-para-todos sem este controle.



(a) Latência (b) Total de mensagens

Figura 5. Latência e total de mensagens para a falha do emissor (fonte).

6. Conclusão

Este trabalho apresentou uma solução distribuída para a difusão confiável (*reliable broadcast*) em sistemas distribuídos assíncronos sujeitos a falhas de *crash*. Árvores com raiz em cada processo são construídas e mantidas dinamicamente sobre uma topologia de hipercubo virtual denominada VCube. O VCube organiza os processos em *clusters* progressivamente maiores e os conecta através de enlaces confiáveis de forma que, se não há falhas, um hipercubo completo é formado. Em caso de falhas, os processos são reorganizados de forma a manter as propriedades logarítmicas do hipercubo.

Além da prova formal do algoritmo, resultados de simulação comparando a solução proposta com uma abordagem um-para-todos mostram a eficiência do mesmo em cenários com e sem falhas, especialmente para sistemas com mais de 128 processos. O cálculo da árvore sob demanda em cada processo e sem a necessidade de troca de mensagens, somada ao controle de mensagens já transmitidas naquela árvore, diminui a latência e o total de mensagens.

Como trabalhos futuros, pretende-se implementar o algoritmo e testá-lo em uma rede real, como o PlanetLab.

Referências

Bonomi, S., Del Pozzo, A. e Baldoni, R. (2013). Intrusion-tolerant reliable broadcast. Technical report, Sapienza Università di Roma,.

Duarte, Jr., E. P., Bona, L. C. E. e Ruoso, V. K. (2014). VCube: A provably scalable distributed diagnosis algorithm. In: *5th Work. on Latest Advances in Scalable Algorithms for Large-Scale Systems*, ScalA'14, pp. 17–22, Piscataway, USA. IEEE Press.

Eugster, P. T., Guerraoui, R., Handurukande, S. B., Kouznetsov, P. e Kermarrec, A.-M. (2003). Lightweight probabilistic broadcast. *ACM Trans. Comput. Syst.*, 21(4):341–374.

Freiling, F. C., Guerraoui, R. e Kuznetsov, P. (2011). The failure detector abstraction. *ACM Computing Surveys*, 43:9:1–9:40.

Garcia-Molina, H. e Kogan, B. (1988). An implementation of reliable broadcast using an unreliable multicast facility. In: *SRDS'98*, pp. 101–111.

Hadzilacos, V. e Toueg, S. (1993). Fault-tolerant broadcasts and related problems. In: Distributed systems, pp. 97–145. ACM Press, New York, NY, USA, 2 ed.

Jeanneau, D., Rodrigues, L. A., Arantes, L. e Duarte, E. P. (2016). An autonomic hierarchical reliable broadcast protocol for asynchronous distributed systems with failure detector. In: *7th Latin-American Symp. Dep. Comput. (LADC)*, pp. 91–98.

Kshemkalyani, A. D. e Singhal, M. (2008). *Message ordering and group communication*, In: Distributed Computing: Principles, Algorithms, and Systems. Cambridge University Press, New York, NY, USA, 1 ed.

Leitão, J., Pereira, J. e Rodrigues, L. (2007). HyParView: A membership protocol for reliable gossip-based broadcast. In: *DSN*, pp. 419–429.

Liebeherr, J. e Beam, T. (1999). HyperCast: A protocol for maintaining multicast group members in a logical hypercube topology. In: Rizzo, L. e Fdida, S., editores, *Networked Group Communication*, v. 1736 de *LNCS*, pp. 72–89. Springer Berlin Heidelberg.

Pereira, J., Rodrigues, L., Pinto, A. e Oliveira, R. (2004). Low latency probabilistic broadcast in wide area networks. In: *SRDS'04*, pp. 299–308.

Ramanathan, P. e Shin, K. (1988). Reliable broadcast in hypercube multicomputers. *IEEE Trans. Comput.*, 37(12):1654–1657.

Rodrigues, L. A., Duarte Jr., E. P. e Arantes, L. (2014). Árvores geradoras mínimas distribuídas e autonômicas. In: *XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, SBRC'14.

Schneider, F. B., Gries, D. e Schlichting, R. D. (1984). Fault-tolerant broadcasts. *Sci. Comput. Program.*, 4(1):1–15.

Urbán, P., Défago, X. e Schiper, A. (2002). Neko: A single environment to simulate and prototype distributed algorithms. *Journal of Inf. Science and Eng.*, 18(6):981–997.

Wu, J. (1996). Optimal broadcasting in hypercubes with link faults using limited global information. *J. Syst. Archit.*, 42(5):367–380.

Yang, Z., Li, M. e Lou, W. (2009). R-code: Network coding based reliable broadcast in wireless mesh networks with unreliable links. In: *GLOBECOM'09*, pp. 1–6.

VCube-PB: Uma Solução Publish/Subscribe Autonômica e Escalável com Difusão Confiável Causal

João Paulo de Araujo[1], **Luiz A. Rodrigues**[2], **Luciana Arantes**[1], **Elias P. Duarte Jr.**[3] **e Pierre Sens**[1]

[1] Sorbonne Universités, UPMC, CNRS, Inria, França

[2]Colegiado de Ciência da Computação – Univ. Estadual do Oeste do Paraná, Brasil

[3]Departamento de Informática – Univ. Federal do Paraná, Brasil

joao.araujo@lip6.fr, luiz.rodrigues@unioeste.br, luciana.arantes@lip6.fr, elias@inf.ufpr.br, pierre.sens@lip6.fr

Abstract. *This paper presents VCube-PB, a topic-based Publish/Subscribe (Pub/Sub) solution implemented on top of VCube, a virtual hypercube-like topology that organizes the processes based on their states (faulty or fault-free). Any published message regarding a topic is sent to subscribers through a spanning tree built on the hypercube with root in the publisher of the message. Messages are propagated using reliable broadcast and delivered to the subscribers respecting the causal order. Experiments with different scenarios were conducted on top of the simulator PeerSim and the results confirm the scalability of the proposed solution, as well as its good performance in dynamic scenarios.*

Resumo. *Este trabalho apresenta VCube-PB, uma solução de Publish/Subscribe (Pub/Sub) baseada em tópicos implementada sobre o VCube, uma topologia virtual baseada em hipercubo que organiza os processos do sistema com base nos seus estados (falho ou sem falha). Toda mensagem publicada referente a um tópico é enviada aos assinantes (subscribers) interessados neste tópico por meio de uma árvore geradora hierárquica criada dinamicamente sobre o hipercubo com raiz no nodo editor (publisher) da mensagem. A difusão é confiável e o sistema de Pub/Sub garante a entrega das mensagens respeitando a precedência causal entre elas. Experimentos com o simulador PeerSim foram realizados com diferentes cenários e os resultados confirmam a escalabilidade da solução proposta, bem como seu bom desempenho em cenários com dinamicidade.*

1. Introdução

Sistemas *Publish/Subscribe* (*Pub/Sub*) são caracterizados por uma rede de nodos distribuídos em que um ou mais nodos editores (*publishers*) produzem mensagens (eventos) e nodos assinantes (*subscribers)*, as consomem [Baldoni et al. 2005, Esposito et al. 2013]. A comunicação entre editores e assinantes é coordenada pelos próprios nodos, garantindo a entrega das mensagens produzidas a todos os assinantes nelas interessados de forma assíncrona.

Existem basicamente dois modelos de sistemas de *Pub/Sub*: os baseados em tópicos (*topic-based*) e os baseados em conteúdo (*content-based*) [Eugster et al. 2003]. No modelo em tópicos, as mensagens são classificadas em assuntos pré-definidos. Exemplos

de tópicos seriam "SBRC 2017" e "Previsão do tempo em Belém". Um assinante pode se cadastrar em quantos tópicos quiser e irá receber todas as mensagens relacionadas aos tópicos em que se registrou. Já no modelo baseado em conteúdo, os eventos são estruturados na forma de atributos múltiplos. O assinante deve, portanto, explicitar em quais atributos e valores ele tem interesse. Um exemplo seria: "temperatura: $< 30\,^{\circ}\mathrm{C}$","local = Belém", "mês = maio". A vantagem do primeiro modelo é que a classificação das mensagens em tópicos é estática, a difusão de mensagens aos assinantes é geralmente baseada em grupos *multicast* e, a interface oferecida ao usuário é simples. Por outro lado, o modelo em conteúdo oferece uma maior flexibilidade ao assinante para poder definir os eventos que lhe interessam, porém uma interface de usuário mais complexa.

Neste trabalho, estamos interessados em sistemas de *Pub/Sub* por tópicos e, particularmente, em oferecer uma forma eficiente, confiável e escalável para a difusão de mensagens entre assinantes de um tópico e a garantia de entrega destas mensagens respeitando a ordem causal entre elas.

No sistema proposto, denominado *VCube-PB*, um assinante registra ou remove a assinatura para o tópico t invocando as funções SUBSCRIBE(t) ou UNSUBSCRIBE(t), respectivamente. Um nodo publica uma mensagem m associada ao tópico t, invocando a função PUBLISH(t, m). Inspirado em Rodrigues et al. (2014), a mensagem m é transmitida de forma confiável (*reliable broadcast*) a todos os assinantes não falhos de t por meio de uma árvore geradora (*spanning tree*), cuja raiz é o nodo editor de m. Esta árvore é construída dinamicamente pelo módulo denominado *VCube-Top*. Trata-se de uma extensão do sistema de diagnóstico *VCube* [Duarte et al. 2014], que organiza todos os nodos do sistema em uma topologia virtual em hipercubo e, a cada rodada de testes de monitoração, informa os nodos da composição do sistema, considerando que nodos podem falhar. O *VCube* apresenta importantes propriedades logarítmicas, mesmo quando processos falham. No caso do *VCube-Top*, existem múltiplos hipercubos lógicos pois os nodos são logicamente organizados em um hipercubo por tópico, não necessariamente completo. Estes se reorganizam automaticamente quando há novos registros ou cancelamentos de assinaturas para o tópico em questão ou quando falhas de nodos são detectadas. Observe que o sistema é considerado autonômico, pois nodos se monitoram e automaticamente se adaptam às mudanças. Enfatizamos também que as modificações na composição dos assinantes de um tópico são transmitidas aos demais assinantes e editores somente a cada nova rodada de testes do *VCube-Top*. Além disso, a precedência causal entre mensagens relacionadas a um mesmo tópico é respeitada. Para implementar tal ordem causal, utilizamos o princípio de "barreira causal" (*causal barrier*) [Raynal 2013], em função da sua escalabilidade.

Vale ressaltar que existem sistemas de *Pub/Sub* na literatura baseados em árvores de difusão [Castro et al. 2006, Zhuang et al. 2001, Gao et al. 2011]. No entanto, a árvore gerada por estes sistemas é fixa, a sua manutenção por vezes custosa e, toda difusão de uma nova publicação é realizada por um único nodo raiz que, consequentemente, pode se tornar um gargalo para o bom desempenho do sistema. No *VCube-PB*, árvores geradoras são criadas dinamicamente a cada publicação de mensagem, não tendo custo de manutenção mesmo em caso de falhas ou mudança na composição dos assinantes/editores e tendo como raiz não um mesmo nodo fixo, mas o nodo editor da mensagem. Um outro ponto a assinalar é que poucos sistemas de *Pub/Sub* existentes na literatura (por exemplo, JEDI [Cugola et al. 2001]) garantem a precedência causal entre mensagens.

O restante do texto está organizado nas seguintes seções. A Seção 2 descreve trabalhos relacionados. A Seção 3 define o modelo do sistema e a interface em termos das funções. Discutimos, na Seção 4, a difusão causal confiável de publicações num contexto dinâmico de assinaturas. Na Seção 5, os sistemas *VCube* e *VCube-Top* são apresentados, assim como os algoritmos propostos para o *VCube-PB*. A Seção 6 contém os resultados da avaliação experimental, enquanto a Seção 7 conclui o trabalho.

2. Trabalhos Relacionados

Scribe [Castro et al. 2006] é um sistema de *Pub/Sub* baseado em tópicos em que toda nova assinatura é roteada por meio do *overlay peer-to-peer* Pastry [Rowstron e Druschel 2001] até encontrar um outro assinante, criando assim uma árvore geradora. Todos os eventos são propagados pela árvore a partir da raiz do tópico, que é fixa. De forma similar, Bayeux [Zhuang et al. 2001] utiliza o *overlay* Tapestry para implementar o serviço de inscrição e disseminação de mensagens a partir do nodo editor. Marshmallow [Gao et al. 2011] é uma solução baseada em conteúdo que também utiliza um modelo de inscrição agrupado em árvores sobre o *overlay* Pastry. Quando um nodo deseja publicar uma mensagem, ele a encaminha para o nodo raiz responsável pelo atributo correspondente, que a propaga por meio da árvore *multicast*.

TIBCO Rendezvous [TIBCO 2016] utiliza uma abordagem distribuída em uma arquitetura *peer-to-peer*. No caso de rede local, mensagens são propagadas por meio de *broadcast* UDP e as mensagens são filtradas por tópico localmente em cada nodo assinante. Gryphon [Strom et al. 1998], Siena [Carzaniga et al. 2001], JEDI [Cugola et al. 2001] e Hermes [Pietzuch e Bacon 2002] utilizam uma abordagem híbrida, com múltiplos servidores (*brokers*) distribuídos, responsáveis por garantir os requisitos da aplicação (consistência, confiabilidade, disponibilidade, etc.) e filtragem/roteamento das mensagens. Por exemplo, no JEDI, a disseminação de eventos/mensagens é feita por meio de uma árvore de servidores e clientes podem se conectar em qualquer nodo da árvore. A ordenação causal dos eventos é garantida. No entanto, essa arquitetura estática sobrecarrega os servidores raiz e causa interrupção do serviço em caso de falha dos servidores intermediários.

Google Pub/Sub [Google 2016] é um serviço baseado em tópicos que visa à alta disponibilidade e escalabilidade utilizando múltiplos servidores de forma balanceada. A solução é utilizada por diversos aplicativos da empresa, como *Ads* e *Gmail*, mas também pode ser contratado por terceiros. GraPS [Canas et al. 2015] utiliza um modelo baseado em grafos. Pontos de interesse são mapeados em vértices e as arestas representam o relacionamento entre pontos.

Outras soluções de *Pub/Sub* se encontram nos *surveys* [Baldoni et al. 2005], [Eugster et al. 2003], [Hinze e Buchmann 2010] e [Esposito et al. 2013].

3. Modelo do Sistema e Funções Oferecidas

Considera-se um sistema distribuído composto por um conjunto finito $\Pi = \{p_0, .., p_{n-1}\}$ com $n = 2^d$ processos, $d > 0$, que se comunicam por troca de mensagens. Cada processo está alocado em um nodo distinto. Assim, os termos *nodo* e *processo* são utilizados com o mesmo sentido.

A rede é representada por um grafo completo com enlaces confiáveis: cada par de processos está conectado diretamente por enlaces ponto-a-ponto bidirecionais e mensagens trocadas entre dois processos nunca são perdidas, corrompidas ou duplicadas. O

sistema admite falhas por parada (*crash*) e estas são permanentes. Um processo que nunca falha é denominado *correto*. Caso contrário, ele é *falho*. O sistema é considerado síncrono, ou seja, os atrasos máximos para a velocidade de execução de processos e transmissão de mensagens são conhecidos.

Funções disponibilizadas à aplicação: um nodo registra e remove seu interesse em um determinado tópico t, isto é, se torna ou deixa de ser assinante de t, invocando as funções SUBSCRIBE(t) e UNSUBSCRIBE(t) respectivamente. Um nodo registra e remove seu interesse em publicar mensagens para um tópico t invocando as funções PUBLISHER_REGISTER(t) e PUBLISHER_UNREGISTER(t) respectivamente. Estas funções são necessárias para que o nodo editor receba ou cesse de receber as modificações referentes à composição do grupo dos assinantes de t. Tendo se registrado como editor para o tópico t, um nodo pode publicar uma mensagem m invocando a função PUBLISH(t, m). Note que um nodo editor de um tópico pode ser ou não assinante deste tópico, o que caracteriza um grupo *multicast* aberto [Raynal 2013].

4. Difusão Confiável Baseada em Tópicos com Ordenação Causal

Sem perda de generalidade e por questões de legibilidade dos algoritmos, definiu-se a função TOPIC_TAG (t), aplicada a cada tópico. Essa função define um valor exclusivo de identificação do tópico t dentro do intervalo [1, .., MAX_TOPICS]. O valor de "*topic_tag*" é incluído nas mensagens do sistema *Pub/Sub*.

No sistema *VCube-PB*, a publicação de uma mensagem relacionada a um tópico é feita por meio de um algoritmo de difusão que garante, **por tópico**, uma difusão causal confiável (*reliable causal broadcast*). Um nodo correto que assina o tópico t é considerado um **assinante correto** após a primeira entrega de uma mensagem de t até o momento em que ele remove seu interesse por t ou falha. As seguintes propriedades garantem a confiabilidade do algoritmo de difusão:

- **Validade** (*validity*): se um processo correto i publica por meio de difusão uma mensagem m, i também entregará m em um tempo finito;
- **Integridade** (*integrity*): toda mensagem m é entregue por um processo no máximo uma vez (não-duplicação) e somente se a difusão de m foi realizada anteriormente (não-criação);
- **Acordo** (*agreement*): se um processo correto entrega a mensagem m referente ao tópico t, então todo assinante correto de t entrega m em um tempo finito.

Por uma questão de coerência, o algoritmo de difusão garante também a ordem causal entre as mensagens publicadas para um mesmo tópico. Assim, se um processo correto entrega m e depois publica m', todo processo correto, assinante de t, entrega m' somente depois de ter entregue m. No entanto, nodos entram e saem dinamicamente do sistema *Pub/Sub* proposto, pois um nodo não é necessariamente um assinante de um tópico desde o início da execução do sistema e, uma vez assinado em um ou mais tópicos, pode cancelar as inscrições. Portanto, a causalidade entre a difusão de duas mensagens somente pode ser satisfeita para um dado assinante, se este último já estivesse no sistema quando da difusão da primeira mensagem.

Ordem causal: se um processo publica uma mensagem m' após uma mensagem m ter sido entregue para ele, nenhum outro processo entregará m depois de m'.

Para implementar a causalidade no algoritmo de difusão do sistema *Pub/Sub*, utilizaremos o princípio de *barreira causal* [Raynal 2013]. A vantagem desta abordagem

é que o controle da causalidade não é baseado na identificação dos nodos, mas sim na dependência direta de mensagens, o que torna o algoritmo adequado à dinamicidade dos nodos (assinaturas, cancelamento de assinaturas ou falha de nodos).

Sejam m e m' duas mensagens da aplicação. A mensagem m **precede diretamente** a mensagem m' (denotado por $m \prec_{im} m'$) se a difusão de m precede causalmente a difusão de m' e não existe m'' tal que a difusão de m preceda causalmente a difusão de m'' e esta preceda causalmente a difusão de m'.

A barreira causal de m (cb_m) consiste no conjunto de mensagens que precedem diretamente m. A Figura 1 retrata a difusão de mensagens em uma rede em que, inicialmente, três nodos (p_0, p_1 e p_2) estão presentes e são assinantes corretos de um tópico t. À esquerda está uma visão do envio e recebimento das mensagens ao longo do tempo e, à direita, o grafo com as dependências entre as mensagens. No exemplo, a entrega de m_3 está condicionada à entrega anterior de m_1 ($m_1 \prec_{im} m_3$), visto que p_1 recebeu m_1 antes de iniciar a difusão de m_3 ($cb_{m_3} = \{m_1\}$). Já m_4 só pode ser entregue após a entrega de m_2 e m_3 pois $cb_{m_4} = \{m_2, m_3\}$. Observe que como m_1 é uma dependência indireta de m_4 (já que m_1 precede m_3), ela não é incluída na barreira causal de m_4. Ainda na Figura 1, o nodo p_3 entra no sistema depois que as mensagens m_2 e m_3 foram difundidas. No entanto, p_3 pode entregar m_4 pois ele ainda não é considerado um assinante correto de t. Apenas após essa primeira entrega, p_3 torna-se um assinante correto e passa a respeitar a ordem causal.

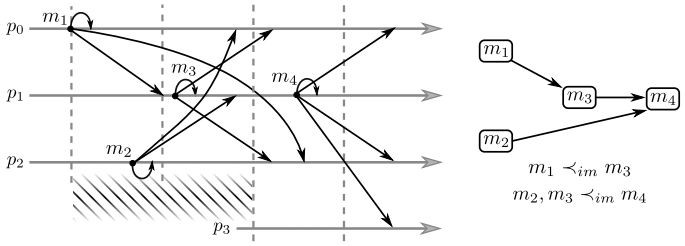


Figura 1. Barreira Causal.

5. Implementação do Sistema VCube-PB

Esta seção apresenta, inicialmente, um resumo breve do *VCube*. Em seguida, descrevemos como adaptamos o princípio do *VCube* para organizar assinantes de um mesmo tópico em diferentes topologias de hipercubo virtual (*VCube-Top*). Por fim, descrevemos os algoritmos utilizados pelo *VCube-PB* para publicação e entrega de mensagem.

5.1. VCube

O *VCube* é uma topologia baseada em hipercubo virtual de dimensão d, criada e mantida com base nas informações de diagnóstico obtidas por meio de um sistema de monitoramento de processos descrito em Duarte Jr. et al. (2014). No *VCube*, cada processo testa outros processos no sistema para verificar se estão corretos ou falhos. Um processo é considerado *correto* se a resposta ao teste for recebida corretamente dentro do intervalo de tempo esperado. Caso contrário, o processo é considerado *falho*. Como o sistema é síncrono, a detecção de falhas é perfeita, ou seja, não há falsas suspeitas.

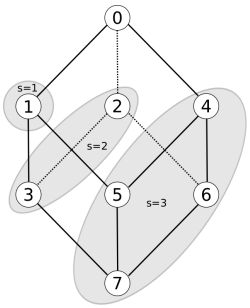
Os processos são organizados em *clusters* progressivamente maiores. Cada *cluster* $s = 1, .., \log_2 n$ possui 2^{s-1} elementos, sendo n o total de processos no sistema. Os testes

são executados em rodadas. Para cada rodada, um processo i testa o primeiro processo sem-falha j na lista de processos de cada *cluster* s e obtém de j as informações que este possui sobre os demais processos do sistema.

Seja $\oplus$ o símbolo que representa a operação binária OU-Exclusivo (XOR). Os membros de cada *cluster* s e a ordem na qual eles são testados por um processo i são obtidos da lista gerada pela seguinte função $c_{i,s}$:

$$c_{i,s} = (i \oplus 2^{s-1}, c_{i\oplus 2^{s-1},1}, ..., c_{i\oplus 2^{s-1},s-1})$$

A Figura 2 exemplifica a organização hierárquica dos processos em um hipercubo de três dimensões com $n = 2^3$ elementos. A tabela da direita contém os elementos de cada *cluster* $c_{i,s}$. Como exemplo, na primeira rodada de testes, o processo p_0 testa: p_1, primeiro processo no *cluster* $c_{0,1} = (1)$, o processo p_2, primeiro processo no *cluster* $c_{0,2} = (2, 3)$ e, por fim, p_4 do *cluster* $c_{0,3} = (4, 5, 6, 7)$. Cada um destes processos envia a p_0 as informações que possuem sobre o estado dos demais processos do sistema. Como todo processo executa estes procedimentos de forma concorrente, ao final da última rodada todo processo será testado ao menos uma vez por um outro processo, garantindo assim que, em $\log_2^2 n$ rodadas (latência de diagnóstico), todos os processos terão localmente a informação atualizada sobre o estado dos demais processos no sistema.



s	$c_{0,s}$	$c_{1,s}$	$c_{2,s}$	$c_{3,s}$	$c_{4,s}$	$c_{5,s}$	$c_{6,s}$	$c_{7,s}$
1	1	0	3	2	5	4	7	6
2	2 3	3 2	0 1	1 0	6 7	7 6	4 5	5 4
3	4 5 6 7	5 4 7 6	6 7 4 5	7 6 5 4	0 1 2 3	1 0 3 2	2 3 0 1	3 2 1 0

Figura 2. Organização Hierárquica do VCube de $d = 3$ dimensões com os *clusters* do processo 0 (esquerda) e a tabela completa da $c_{i,s}$ (direita).

5.2. VCube-Top: Estendendo o VCube para Implementar o Sistema Pub/Sub

No sistema de *Pub/Sub* proposto, os $n = 2^d$ nodos que compõem o sistema não são posicionados em um único hipercubo como no *VCube* original, mas em múltiplos, ou seja, os nodos são logicamente e dinamicamente organizados em uma topologia de hipercubo por tópico e esta não é necessariamente completa.

O nodo i adere ou retira-se do hipercubo referente ao tópico t, como assinante e/ou editor, invocando respectivamente as funções VCUBE_JOIN(t, $tipo$) ou VCUBE_LEAVE (t, $tipo$), oferecidas pelo *VCube-Top*, sendo $tipo$ igual a $subscriber$ ou $publisher$. Note que contrariamente ao *VCube*, um mesmo nodo pode participar de diferentes hipercubos e integrar/retirar-se de um mesmo hipercubo inúmeras vezes. Novas inscrições e cancelamentos de assinaturas de um nodo a um tópico são armazenados localmente e na próxima rodada de monitoramento do *VCube-Top*, as modificações na composição dos assinantes deste tópico são difundidas aos demais assinantes. O conjunto de mudanças é então notificado à camada superior. As seguintes funções definidas em [Rodrigues et al. 2014] foram também estendidas para incluir o tópico:

- $cluster_i(j) = s$: determina a qual *cluster* s do processo i o processo j pertence.
- $FF\_neighbor_i(t, s) = j$: retorna o primeiro processo j correto do *cluster* $c_{i,s}$ que assinou o tópico t. Se todos os processos de $c_{i,s}$ estão falhos ou não são assinantes de t, a função retorna $\perp$.
- $neighborhood_i(t, k) = \{j \mid j = FF\_neighbor_i(t, s), 1 \leq s \leq k\}$: gera, para o tópico t, um conjunto que contém todos os processos sem-falha virtualmente conectados ao processo i de acordo com $FF\_neighbor_i(t, s)$, para $s = 1, .., k$. Por exemplo, no hipercubo sem falhas da Figura 2 se todos os nodos são assinantes de t, $neighborhood_0(t, 3) = \{1, 2, 4\}$, $neighborhood_0(t, 2) = \{1, 2\}$, $neighborhood_4(t, 2) = \{5, 6\}$, etc.

5.3. Algoritmos do VCube-PB

O Algoritmo 5.1 contém as funções oferecidas à aplicação. Um nodo que deseja se registrar apenas como assinante de um tópico t invoca a função SUBSCRIBE(t). Para deixar de receber as mensagens do referido tópico invoca UNSUBSCRIBE(t). Antes de iniciar a publicação no tópico t, o nodo deve se inscrever no mesmo invocando PUBLISH_REGISTER(t), que o adiciona na lista $ptopics_i$. Para publicar uma mensagem m relacionada ao tópico t, basta invocar função PUBLISH(t, m). Essa, por sua vez, após verificar que o nodo está registrado como editor do tópico (linha 14), invoca o procedimento CO_BROADCAST(m) do Algoritmo 5.2. Além dessa função, o Algoritmo 5.2 inclui os procedimentos responsáveis pelo encaminhamento de m a todos os assinantes de t por meio da árvore de difusão do tópico e, a função CO_DELIVER(m) que entrega m à aplicação, assegurando a ordem causal entre as mensagens deste mesmo tópico. O Algoritmo 5.3 contém funções e procedimentos auxiliares utilizados pelo Algoritmo 5.2. Detalhes desse algoritmo serão omitidos por questões de espaço.

Algoritmo 5.1 Interface do VCube-PB no nodo i

```
1:  procedure SUBSCRIBE(topic t)
2:      VCUBE_JOIN(t,"subscriber")
3:      stopics_i ← stopics_i ∪ {t}

4:  procedure UNSUBSCRIBE(topic t)
5:      VCUBE_LEAVE(t,"subscriber")
6:      stopics_i ← stopics_i ∖ {t}

7:  procedure PUBLISHER_REGISTER(topic t)
8:      VCUBE_JOIN(t,"publisher")
9:      ptopics_i ← ptopics_i ∪ {t}

10: procedure PUBLISHER_UNREGISTER(topic t)
11:     VCUBE_LEAVE(t,"publisher")
12:     ptopics_i ← ptopics_i ∖ {t}

13: function PUBLISH(topic t, message m)
14:     if t ∉ ptopics_i then
15:         return NOK
16:     m.source ← i
17:     m.topic ← topic_tag(t)
18:     m.counter ← counter_i[t]
19:     counter_i[t] ← counter_i[t] + 1
20:     CO_BROADCAST(m)
21:     return OK
```

Atualização da composição dos assinantes: todo processo armazena a composição atual do grupo referente a cada tópico em que é assinante ou editor na variável local $members[MAX\_TOPICS]$, que é indexada por tópico. Assim, a cada rodada de testes do *VCube-Top*, este informa (*upon notifying memeberhip* ou *upon notifying crash*, Algoritmo 5.2) as últimas modificações referentes à composição dos *assinantes* dos tópicos: novos registros, cancelamentos ou falhas de nodos.

Difusão de mensagem: cada mensagem m publicada por um nodo i (ver função PUBLISH(t,m), Algoritmo 5.1) é identificada por um *timestamp* único, que contém a identidade do editor ($source$), o tópico ($topic$) associado e um número de sequência ($counter$). A função CO_BROADCAST(m) adiciona a mensagem m em uma fila indexada

Algoritmo 5.2 Difusão causal confiável executada em um processo i

```
 1: upon notifying MEMBERSHIP(join, leave)
 2:     for all ⟨j,t⟩ ∈ join do
 3:         members_i[t] ← members_i[t] ∪ {j}
 4:         stopics_i ← stopics_i ∪ {t}
 5:     for all ⟨j,t⟩ ∈ leave do
 6:         CHECKLEAVE(j,t)

 7: procedure CO_BROADCAST(message m)
 8:     t ← topic(m)
 9:     if counter(m) = 1 then
10:         create Task T1(t)
11:     queue_i[t] ← queue_i[t] ∪ {m}

12: Task T1(topic t)
13: loop
14:     m ← queue_i[t].first()          ▷ remoção bloqueante
15:     s ← source(m); t ← topic(m); c ← counter(m)
16:     wait until acks_i ∩ ⟨⊥,*,⟨s,t,c−1⟩⟩ = ∅
17:     msgs_i ← msgs_i ∪ {m}
18:     if ∄⟨s,*⟩ ∈ first_rec_i[t] then
19:         first_rec_i[t] ← first_rec_i[t] ∪ {⟨s,c⟩}
20:     CO_DELIVER(m)
21:     delvs_i[t] ← delvs_i[t] ∖ {⟨s,c−1⟩} ∪ {⟨s,c⟩}
22:     m.cb ← causal_barrier_i[t]
23:     for all j ∈ neighborhood_s(t, log_2 n) do
24:         acks_i ← acks_i ∪ {⟨⊥,j,⟨s,t,c⟩⟩}
25:         SEND(⟨TREE,m⟩) to p_j
26:     causal_barrier_i[t] ← {⟨s,c⟩}

27: upon RECEIVE ⟨ACK,⟨s,t,c⟩⟩ from p_j
28:     k ← x : ⟨x,j,⟨s,t,c⟩⟩ ∈ acks_i
29:     acks_i ← acks_i ∖ {⟨k,j,⟨s,t,c⟩⟩}
30:     if k ≠ ⊥ then
31:         CHECKACKS(k, ⟨s,t,c⟩)

32: upon RECEIVE ⟨TREE,m⟩ from p_j
33:     s ← source(m); t ← topic(m)
34:     c ← counter(m); cb ← causal_barrier(m)
35:     if ∄⟨s,*⟩ ∈ first_rec_i[t] then
36:         first_rec_i[t] ← first_rec_i[t] ∪ {⟨s,c⟩}
37:     if (⟨s,c,cb⟩ ∉ recs_i[t])
          and (∄⟨s,c'⟩ ∈ delvs_i[t] : c' ≥ c) then
38:         recs_i[t] ← recs_i[t] ∪ {⟨s,c,cb⟩}
39:         msgs_i ← msgs_i ∪ {m}
40:         CHECKRECS(t)
41:         if s ∉ members_i[t] then
42:             CO_BROADCAST(m)
43:             return
44:     for all k ∈ neighborhood_i(t, cluster_i(j) − 1) do
45:         if ⟨j,k,⟨s,t,c⟩⟩ ∉ acks_i then
46:             acks_i ← acks_i ∪ {⟨j,k,⟨s,t,c⟩⟩}
47:             SEND(⟨TREE,m⟩) to p_k
48:     CHECKACKS(j, ⟨s,t,c⟩)

49: upon notifying CRASH(j)
50:     for all t' ∈ stopics_i do
51:         CHECKLEAVE(j,t')
```

por tópico. Para cada tópico que um nodo é editor existe também uma tarefa $T1$ associada. Essa tarefa inclui em m o conjunto cb, composto pelas mensagens, identificadas por $\langle source, counter\rangle$, que representam a barreira causal de m. Havendo mensagens na fila para t, $T1(t)$ realiza uma difusão FIFO confiável por meio de uma árvore geradora construída dinamicamente e hierarquicamente, com raiz em i, sobre o hipercubo virtual composto pelos membros interessados no tópico associado à mensagem. Para tanto, as funções descritas na Seção 5.2 são utilizadas. Ao invocar a função $neighborhood_i(t, log_2 n)$, o processo i recebe um conjunto com a identificação dos primeiros processos assinantes de t, não-falhos, de cada um de seus *clusters*, que passam a ser considerados como seus filhos na árvore, e envia então a mensagem m a estes (linha 23). Ao receber m, todo filho j de i envia m aos processos $neighborhood_j(t, cluster_j(i) - 1)$, $cluster_j(i) > 1$, (linha 44) e assim sucessivamente.

Por exemplo, considere a Figura 2 sem nodos falhos e que todos os nodos são assinantes do tópico t_1. Nodo p_0 publica uma mensagem m referente a t_1, se tornando assim a raiz da árvore geradora. A mensagem será enviada aos filhos de p_0 : $FF\_neighbor_0(t_1, 1) = 1$, $FF\_neighbor_0(t_1, 2) = 2$ e $FF\_neighbor_0(t_1, 3) = 4$. Ao receber m, p_1 não a retransmite visto que $cluster_1(0) = 1$. O nodo p_2 recebe m ($cluster_2(0) = 2$) e a retransmite a seu filho p_3, o primeiro nodo do seu *cluster* 1 ($c_{2,1}$). Quando p_3 recebe m, como $cluster_3(2) = 1$, p_3 não retransmite m. No caso de p_4 ($cluster_4(0) = 3$), m é recebida e retransmitida aos seus filhos $p_5 \in c_{4,1}$ e $p_6 \in c_{4,2}$. Finalmente, sendo $cluster_6(0) = 2$, o nodo p_6 envia a mensagem para o processo p_7. Considere agora um segundo exemplo na mesma figura em que apenas os nodos $\{p_0, p_2, p_4, p_5, p_6\}$ são assinantes do tópico t_2 e que p_2 publica a mensagem m' referente

Algoritmo 5.3 Funções Auxiliares

```
1:  function CHECKCB(topic t, causal barrier cb)
2:      for all ⟨s, c⟩ ∈ cb do
3:          if (∃ ⟨s', c'⟩ ∈ delvs_i[t]: s = s' and c' ≥ c) or ( ∃ ⟨s', c'⟩ ∈ first_rec_i[t]: s = s' and c' > c) then
4:              cb ← cb ∖ {⟨s, c⟩}
5:      return (cb = ∅)

6:  procedure CHECKRECS(topic t)
7:      repeat
8:          delMsg ← 0
9:          for all s ← s', c ← c', cb ← cb' : ⟨s', c', cb'⟩ ∈ recs_i[t] do
10:             if CHECKCB(t, cb) then
11:                 recs_i[t] ← recs_i[t] ∖ {⟨s, c, cb⟩}
12:                 delvs_i[t] ← delvs_i[t] ∪ {⟨s, c⟩}
13:                 CO_DELIVER(m), m ∈ msgs_i : source(m) = s, topic(m) = t, counter(m) = c
14:                 causal_barrier_i[t] ← causal_barrier_i[t] ∖ cb ∪ {⟨s, c⟩}
15:                 if ∃ m' ∈ msgs_i: source(m') = s and topic(m') = t and #(m') > c then
16:                     msgs_i ← msgs_i ∖ {m}
17:                 delMsg ← delMsg + 1
18:     until (delMsg = 0)

19: procedure CHECKACKS(process j, ⟨s, t, c⟩)
20:     if acks ∩ ⟨j, ∗, ⟨s, t, c⟩⟩ = ∅ then
21:         if j ∈ members_i[t] then
22:             SEND(⟨ACK, ⟨s, t, c⟩⟩) to p_j

23: procedure CHECKLEAVE(process j, topic t)
24:     members_i[t] ← members_i[t] ∖ {j}
25:     if members_i[t] = ∅ then
26:         stopics_i ← stopics_i ∖ {t}
27:         return
28:     first_rec_i[t] ← ∅
29:     k ← FF_neighbor_i(t, cluster_i(j))
30:     for all x ← x', y ← y', s ← s', c ← c' : ⟨x', y', ⟨s', t, c'⟩⟩ ∈ acks_i do
31:         if {s, x} ⊈ members_i[t] then
32:             acks_i ← acks_i ∖ ⟨x, y, ⟨s, t, c⟩⟩
33:         else if q = j then
34:             if k ≠ ⊥ and ⟨x, k, ⟨s, t, c⟩⟩ ∉ acks_i then
35:                 acks_i ← acks_i ∪ {⟨x, k, ⟨s, t, c⟩⟩}
36:                 m ← m' ∈ msgs_i: source(m') = s, topic(m') = t, counter(m') = c
37:                 SEND(⟨TREE, m⟩) to p_k
38:             acks_i ← acks_i ∖ {⟨x, j, ⟨s, t, c⟩⟩}
39:             CHECKACKS(x, ⟨s, t, c⟩)
40:     if (c' ← max c : ⟨j, c, ∗⟩ ∈ recs_i[t]) ≠ ⊥ then
41:         m ← m' ∈ msgs_i: source(m') = j, topic(m') = t, counter(m') = c'
42:         CO_BROADCAST(m)
```

a t_2. Neste caso, o nodo p_2 envia m' para o primeiro processo sem falha de cada *cluster*, interessado no tópico t_2: $FF\_neighbor_2(t_1, 1) = \perp$ (não há assinantes de t_2 no *cluster* $c_{2,1}$), $FF\_neighbor_2(t_2, 2) = 0$ e $FF\_neighbor_2(t_1, 3) = 6$ (p_6 é o primeiro assinante sem-falha, interessado em t_2 de $c_{2,3}$). Ao receber m', p_0 não a retransmite à p_1 pois este não é assinante de t_2. O nodo p_6 verifica que, no *cluster* $c_{6,2} = (4, 5)$, p_4 também é assinante de t_2, enviando-lhe assim m'. Por fim, p_4 retransmite m' para p_5, contido no *cluster* $c_{4,1}$, e assinante de t_2.

Como a função CO_BROADCAST garante uma entrega confiável, cada processo, após enviar m a seus filhos, aguarda destes uma confirmação (mensagem tipo ACK). Um processo somente envia um ACK a seu pai depois de ter recebido um ACK de todos os seus filhos (função CHECKACKS, Algoritmo 5.3). Para o controle dos $ACKs$ pendentes, todo processo i utiliza a variável conjunto $acks_i$ em que cada elemento guarda informação sobre a mensagem enviada ($\langle s, t, c\rangle$ sendo s a identidade do nodo editor, t o tópico

e, c o número de sequência), o emissor da mensagem j e a qual nodo k a mensagem foi retransmitida. Os dados da mensagem são armazenados na variável conjunto $msgs_i$. Se o *VCube-Top* enviar uma notificação ao processo i informando que ocorreu uma mudança no conjunto de assinantes do tópico devido à falha de k (*upon notifying crash*) ou cancelamento de sua assinatura ao tópico t (*upon notifying membership*), a função CHECKLEAVE (Algoritmo 5.3) é invocada. Neste caso, se i enviou m ao processo k mas ainda não recebeu o ACK de k, i retransmite m para o próximo processo sem-falha do *cluster* de k que é assinante de t. Considere o exemplo anterior na Figura 2 com os assinantes de t_2, mas que p_4 falhe após o envio de m' pelo nodo p_6. Ao detectar a falha de p_4, p_6 enviará m' a p_5, que se torna o próximo processo sem-falha com interesse em t_2 no *cluster* $c_{6,2}$.

Observe que o editor de uma mensagem envia uma nova mensagem referente ao mesmo tópico t somente após receber um ACK de todos os seus filhos (linha 16 do Algoritmo 5.2), assegurando assim uma difusão FIFO. Ao receber esta nova mensagem, um nodo pode excluir do seu conjunto $msgs_i$ a mensagem anterior do mesmo tópico, emitida por este editor. Além disso, se o editor de m falhar, todo processo correto que detectou a falha irá fazer uma nova difusão de m, através da árvore geradora do editor (linha 42 do Algoritmo 5.2 e linha 42 do Algoritmo 5.3), assegurando assim a propriedade do *acordo* (ver Seção 4).

Entrega de mensagem: Ao receber uma mensagem m de j referente ao tópico t, o processo i inclui m no conjunto $recs_i[t]$ e no conjunto $first\_rec_i[t]$, caso m seja a primeira mensagem recebida de j por i para este tópico. O processo i verifica então todas as mensagens recebidas que podem ser entregues à aplicação. Para tanto, i invoca a função CHECKRECS (Algoritmo 5.3) que recursivamente verifica as dependências causais com o auxílio da função CHECKCB. O processo i entrega, por meio da chamada da função CO_DELIVER (m), todas as mensagens cuja causalidade direta pode ser satisfeita, assim como aquelas cuja causalidade não precisa ser satisfeita, pois i se tornou assinante do tópico t após o envio destas mensagens. Esta verificação é possível com o auxílio do valor armazenado em $first\_rec_i[t]$ e pelo fato da difusão de mensagens por um mesmo editor ser FIFO: se i recebe m com barreira causal cb e $\exists\ \langle source, counter\rangle \in cb$ e $\exists\ \langle source, counter'\rangle \in first\_rec_i[t]$, tal que $counter' > counter$, m pode ser entregue imediatamente sem respeitar a ordem definida em cb. Todas as mensagens entregues à aplicação são transferidas de $recs_i[t]$ para $delvs_i[t]$. Como já descrito, ao receber m, i a retransmite aos seus filhos na árvore geradora em questão.

6. Avaliação Experimental

Nesta seção são apresentados e discutidos os resultados experimentais obtidos em diferentes cenários para o *VCube-PB*. As simulações foram implementadas utilizando o *framework* PeerSim [Montresor e Jelasity 2009] para simulações de sistemas distribuídos massivamente dinâmicos.

Os cenários de testes variam em número de assinantes e editores e também na frequência com que os nodos registram ou cancelam assinaturas em tópicos. Considera-se que nodos não falham e um único tópico. Os testes visam à avaliar a escalabilidade da solução proposta, bem como o efeito da causalidade entre as mensagens e o impacto da dinamicidade devido a novas assinaturas e cancelamentos de assinaturas.

Foram realizadas 40 simulações para cada cenário. Os resultados obtidos para latência variam de acordo com o tempo de transmissão de uma mensagem por salto (*hop*), o

tempo total até o recebimento por um assinante e o tempo de entrega da mensagem às camadas superiores desde a sua criação. A cada salto na árvore de difusão, uma mensagem tem seu tempo de transmissão definido seguindo uma distribuição uniforme entre 10.0 e 20.0 unidades de tempo. Além disso, todo nodo possui uma capacidade de processamento de 1 mensagem por unidade de tempo. Caso o nodo receba múltiplas mensagens em um mesmo momento, uma única é processada por vez, respeitando a ordem de chegada.

Um nodo editor sem dinamicidade de assinaturas: inicialmente, para mostrar as propriedades logarítmicas do *VCube-PB*, são avaliados cenários com um único editor por tópico em cinco diferentes cenários de distribuição dos nodos assinantes. Estes são estaticamente distribuídos no início da simulação. No primeiro cenário, todos os nodos são assinantes. No segundo cenário, apenas os vizinhos do nodo editor são assinantes do tópico. No terceiro, os assinantes correspondem ao grupo formado pelo primeiro nodo de cada nível da árvore de difusão. Por fim, são avaliados dois cenários com distribuição aleatória de 25% e 50% dos nodos como assinantes.

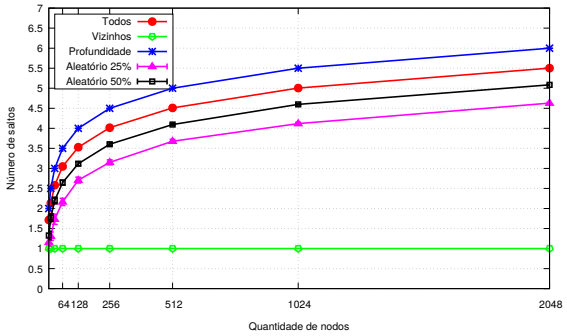


Figura 3. Média de saltos para entrega de uma mensagem a todos os assinantes.

A Figura 3 apresenta a latência média em saltos para que todos os assinantes recebam a mensagem publicada de acordo com os cinco cenários descritos anteriormente. Os dados variam de acordo com o número de nodos no sistemas. Exceto pela curva na qual apenas os vizinhos imediatos do editor estão inscritos, todas as demais possuem um comportamento logarítmico em relação à quantidade de nodos presentes. Isso mostra que a topologia mantém sua característica logarítmica mesmo quando o hipercubo não é completo. Para os cenários mais próximos de uma situação realística, em que os nodos inscritos são escolhidos de forma aleatória, o número médio de saltos é inferior, visto que é possível que a árvore de difusão formada possua menos níveis.

Vários nodos editores com dinamicidade de assinaturas: Contrariamente aos cenários anteriores em que satisfazer a causalidade entre mensagens se resume a entregá-las na ordem FIFO, assegurar a causalidade entre mensagens de diferentes nodos editores faz com que os atrasos na entrega de mensagens sejam mais frequentes e acentuados. Logo, o *overhead* em termos de latência afeta ainda mais o tempo necessário da entrega das mensagens à aplicação.

Nos mesmos cenários acima descritos, o *VCube-PB* foi comparado com uma abordagem com árvore fixa (ex. Scribe [Castro et al. 2006] e Marshmallow [Gao et al. 2011]) na qual o mesmo nodo é sempre a raiz da árvore de difusão, independente do nodo editor. Assim, quando um editor deseja disseminar uma mensagem, ele a envia a esse nodo, que será responsável pela difusão. Isso faz com que todas as mensagens devam necessaria-

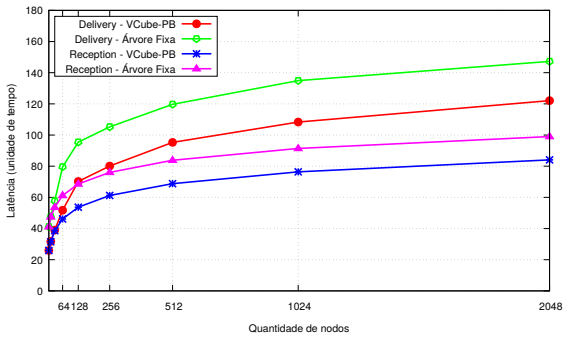
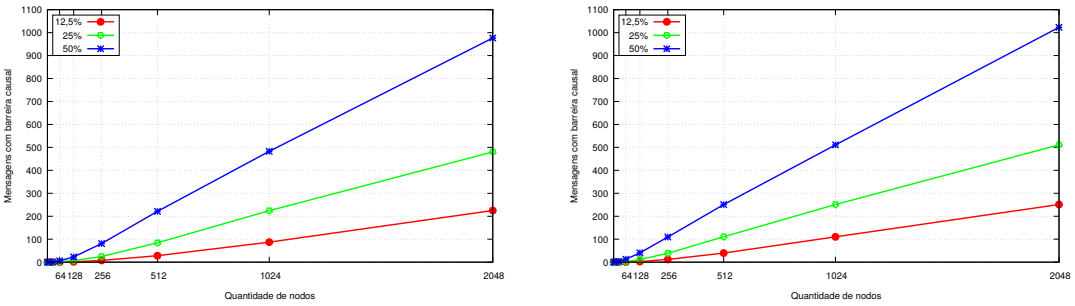


Figura 4. Latência média para recepção e entrega de mensagens.

mente seguir o mesmo caminho e, caso a vazão de mensagens seja alta, a capacidade de processamento do nodo impacta na latência. Por outro lado, para a abordagem proposta com árvores dinâmicas, a posição em que um nodo encontra-se em cada árvore varia de acordo com a sua posição no hipercubo criado pelo *VCube-PB* para o tópico, distribuindo a carga de mensagens, já que cada nodo tem sua árvore de difusão organizada de forma diferente.

A Figura 4 apresenta essa comparação, na qual 25% dos nodos publicam uma única mensagem aleatoriamente em um intervalo de 1000.0 unidades de tempo. As curvas representam as latências médias de recepção e entrega. Também, para aferir a estabilidade da solução proposta em casos de mudança nas assinaturas, a cada 100.0 unidades de tempo ocorre uma variação de até 5% dos nodos inscritos no tópico. Mesmo com o aumento da latência de entrega, podemos observar que o *VCube-PB* apresenta melhores resultados quando comparado com a abordagem de árvore fixa: a latência de entrega do *VCube-PB* tem uma queda entre 37% (8 nodos) e 17% (2048 nodos). Para os resultados exibidos na Figura 4 também foram calculados os intervalos de confiança com nível de confiança em 99%. Como os valores obtidos nunca ultrapassam 4% da média, estes são omitidos da figura.



(a) *VCube-PB* (b) Árvore Fixa

Figura 5. Número de mensagens com dependências causais.

Embora haja uma diminuição da latência quando da utilização de árvores dinâmicas com raiz no nodo editor, o número de mensagens que são afetadas pela causalidade se mantém próximo para as duas abordagens, como retratado nas Figuras 5(a) e 5(b). Em redes com baixo número de nodos, para todos os casos, a probabilidade de que as mensagens tenham seu tempo de entrega afetado pela causalidade é pequena, dado o intervalo

em que são enviadas. Por outro lado, conforme aumenta-se o tamanho da rede, a variação no tempo de cada salto na transmissão das mensagens implica em cada vez mais dependências. Também, ao se saturar a rede com mais nodos publicando em um mesmo intervalo de tempo (50% dos nodos), o número de mensagens afetadas se aproxima do total de mensagens enviadas. O que gera o ganho em termos de latência para a solução proposta é o fato de que o uso de múltiplas árvores distribui a carga do sistema em diferentes partes da rede. Ao se usar uma única árvore por tópico, como o existente no estado da arte, aumenta-se o impacto da capacidade de processamento do nodo na latência de recepção de mensagens. Além disso, o uso de uma única árvore por tópico implica na existência de um ponto de falha no sistema, enquanto que para *VCube-PB*, o editor é raiz de sua própria árvore e, em caso de alterações no conjunto de assinantes, a árvore pode se adaptar mesmo durante a difusão de uma mensagem.

7. Conclusão

Este trabalho apresentou o *VCube-PB*, uma solução distribuída de *publish/subscribe* baseada em tópicos. Mensagens publicadas referentes a cada tópico são transmitidas a todos os assinantes deste tópico por meio de um algoritmo de difusão confiável causal (*reliable causal broadcast*) que utiliza árvores geradoras construídas e reorganizadas dinamicamente sobre a topologia de hipercubo virtual *VCube-Top*, garantindo assim as propriedades logarítmicas do hipercubo mesmo em caso de falhas. Esta também é responsável pela manutenção da visão que um assinante/editor tem dos grupos de assinantes dos tópico nos quais ele também está interessado.

Resultados de avaliação de desempenho sobre o simulador *PeerSim* mostram o comportamento logarítmico do *VCube-PB*, mesmo em situações em que o hipercubo não é completo ou quando há alterações no conjunto de nodos assinantes durante a difusão de mensagens. O uso de múltiplas árvores dinâmicas, embora apresente um número similar de mensagens afetadas pela ordem causal, supera em latência todos os cenários nos quais emprega-se uma única árvore fixa para cada tópico. Como o tempo de processamento das mensagens é um fator limitante no desempenho do sistema, a solução proposta neste trabalho garante uma melhor distribuição da carga de mensagens ao utilizar diferentes árvores de difusão. Além disso, ela elimina o ponto de falha existente ao usar um nodo como raiz fixa para difusão de mensagens em um tópico.

Como trabalhos futuros, além da especificação formal dos algoritmos, serão realizados novos experimentos incluindo cenários com falhas de nodos e uso de *traces* reais.

Agradecimentos

Este trabalho teve apoio do Programa GDE (Doutorado Pleno no Exterior) do CNPq, da Fundação Araucária/SETI (Convênio 144/2015-45112) do projeto 309143/2012-8 do CNPq.

Referências

Baldoni, R., Querzoni, L. e Virgillito, A. (2005). Distributed event routing in Publish/Subscribe communication systems: a survey. Technical report, MIDLAB, Departament of Informatics and Systems, Univerisity of Rome.

Canas, C., Pacheco, E., Kemme, B., Kienzle, J. e Jacobsen, H. (2015). Graps: A graph Publish/Subscribe middleware. Middleware '15, pp. 1–12.

Carzaniga, A., Rosenblum, D. e Wolf, A. (2001). Design and evaluation of a wide-area event notification service. *ACM Trans. Comput. Syst.*, 19(3):332–383.

Castro, M., Druschel, P., Kermarrec, A. M. e Rowstron, A. I. (2006). Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE J.Sel. A. Commun.*, 20(8):1489–1499.

Cugola, G., Nitto, E. D. e Fuggetta, A. (2001). The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. *IEEE Trans. Softw. Eng.*, 27(9):827–850.

Duarte, E., Bona, L. e Ruoso, V. (2014). VCube: A provably scalable distributed diagnosis algorithm. In: *5th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, pp. 17–22.

Esposito, C., Cotroneo, D. e Russo, S. (2013). Survey on reliability in Publish/Subscribe services. *Comput. Netw.*, 57(5):1318–1343.

Eugster, P., Felber, P., Guerraoui, R. e Kermarrec, A. (2003). The many faces of Publish/Subscribe. *ACM Comput. Surv.*, 35(2):114–131.

Gao, S., Li, G. e Zhao, P. (2011). Marshmallow: A content-based publish-subscribe system over structured p2p networks. In: *7th CIS*, pp. 290–294.

Google (2016). CLOUD PUB/SUB: A global service for real-time and reliable messaging and streaming data. Disponível em: https://cloud.google.com/pubsub.

Hinze, A. e Buchmann, A., editores (2010). *Principles and Applications of Distributed Event-Based Systems*. IGI Global.

Montresor, A. e Jelasity, M. (2009). Peersim: A scalable p2p simulator. In: *2009 IEEE Ninth International Conference on Peer-to-Peer Computing*, pp. 99–100.

Pietzuch, P. R. e Bacon, J. (2002). Hermes: A distributed event-based middleware architecture. In: *Proc. 22nd Int'l Conf. Distr. Comp. Sys. (ICDCSW'02)*, pp. 611–618.

Raynal, M. (2013). *Distributed Algorithms for Message-Passing Systems*. Springer Publishing Company, Incorporated.

Rodrigues, L., Arantes, L. e Duarte, E. (2014). An autonomic implementation of reliable broadcast based on dynamic spanning trees. In: *10th EDCC*, pp. 1–12.

Rowstron, A. e Druschel, P. (2001). Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. Middleware '01, pp. 329–350.

Strom, R., Banavar, G., Chandra, T., Kaplan, M., Miller, K., Mukherjee, B., Sturman, D. e Ward, M. (1998). Gryphon: An information flow based approach to message brokering. In: *Proc. 9th Int'l Symp. on Soft. Reliability Eng.*

TIBCO (2016). TIBCO rendezvous messaging middleware. Disponível em: http://www.tibco.com/products/automation/enterprise-messaging/rendezvous.

Zhuang, S., Zhao, B., Joseph, A., Katz, R. e Kubiatowicz, J. (2001). Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. NOSSDAV '01, pp. 11–20.

Trilha Principal do SBRC 2017
Sessão Técnica 6
Roteamento

Um Protocolo de Roteamento para o Consumo Balanceado de Energia em Redes de Sensores Aquáticas

Rodolfo W. L. Coutinho[1,2], Azzedine Boukerche[2], Luiz F. M. Vieira e Antonio A. F. Loureiro[1]

[1]Depto. de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Av. Antonio Carlos, 6627, Belo Horizonte, MG, Brasil

{rwlc, lfvieira, loureiro}@dcc.ufmg.br

[2]School of Electrical Engineering and Computer Science – University of Ottawa
800 King Edward Ave., Ottawa, Canadá

boukerch@site.uottawa.ca

Abstract. *Opportunistic routing (OR) has emerged as a promising paradigm to the design of routing protocols for underwater sensor networks (UWSNs). However, despite of its advantages, it introduces a critical problem that has been neglected in the current proposed protocols for UWSNs: the immutable transmission priority level of the next-hop forwarding nodes, which lead to an overuse of a unique node (or a few of them), quickly depleting its battery and shortening the network lifetime. In this paper, we shed light on the need for mechanisms for rotating the forwarding priority level between candidate nodes. We propose the energy-aware opportunistic routing (EnOR) protocol, which rotates the transmission priority level of the forwarding candidate nodes by considering the remaining energy, link reliability and packet advancement of them. Simulation results reveal that EnOR effectively extends the network lifetime as compared with other underwater sensor network opportunistic routing protocols.*

Resumo. *Protocolos de roteamento oportunístico têm sido extensivamente propostos para redes de sensores sem fio sub-aquáticas. Apesar de seus imensos benefícios, esses protocolos geralmente introduzem um consumo desigual de energia nos nós sensores. Isso acontece por causa da atribuição imutável das prioridades de transmissão nos nós sensores, que faz com que alguns nós participem extensivamente do roteamento de dados e esgotem suas baterias rapidamente, causando partições na rede e, consequentemente, uma diminuição do tempo de vida útil de uma rede de sensores sub-aquática. Este trabalho propõe um novo protocolo de roteamento oportunístico, chamado de EnOR. EnOR rotaciona a prioridade de transmissão dos nós de próximo salto, considerando a energia residual, confiabilidade dos enlaces acústicos e o avanço do pacote em direção ao destinatário. Resultados de simulação mostram que o protocolo proposto, quando comparado com trabalhos relacionados, estende o tempo de vida da rede de sensores sub-aquática.*

1. Introdução

O projeto de protocolos e algoritmos para redes aquáticas é bastante desafiador. Isso é devido as características da comunicação acústica em ambientes aquáticos como, por

exemplo, a baixa largura de banda, alto atraso de propagação, propagação do sinal por múltiplos caminhos, alto ruído, alto custo energético para comunicação sub-aquática, entre outros [Stojanovic and Preisig 2009, Partam et al. 2006]. Tradicionalmente, uma rede de sensores sub-aquática [Akyildiz et al. 2005] é formada por nós sensores sub-aquáticos com capacidades de sensoriamento, processamento e comunicação sem fio. Estes nós são usados para colaborativamente monitorar eventos e variáveis de interesse. Além disso, em uma rede de sensores sub-aquática, um ou diversos sorvedouros (*sinks*) são depositados na superfície. Estes nós são equipados com modems acústicos e rádio, o que lhes permitem comunicação sem fio com nós sensores sub-aquáticos e com a central de monitoramento, por exemplo, um navio. Mesmo com esses desafios, redes de sensores sem fio sub-aquáticas vêm ganhando crescente atenção da comunidade científica e industrial, no Brasil e no mundo. A razão desse crescente interesse é devido aos seus benefícios para aplicações distribuídas de monitoramento e exploração de ambientes sub-aquáticos.

Em redes de sensores sub-aquáticas, protocolos de roteamento oportunísticos têm sido preferencialmente adotados [Yan et al. 2008, Xie et al. 2006, Lee et al. 2010, Noh et al. 2013, Coutinho et al. 2016b], devido ao seu potencial para entrega dos dados. No paradigma de roteamento oportunístico, um subconjunto dos nós vizinhos, chamados de **candidatos**, é determinado. Para isso, o protocolo de roteamento utiliza algum critério, tais como número esperado de transmissões, avanço do pacote em direção ao destinatário ou o atraso médio. Esses nós são aptos a continuarem o encaminhamento do pacote, via múltiplos saltos, até o destinatário. Assim, o nó fonte transmite o pacote de dados, endereçando-o para esse subconjunto dos vizinhos. Os nós selecionados como próximo salto, irão coordenadamente por meio de prioridades, continuar o processo de entrega. Nessa abordagem, como mais de um vizinho é selecionado como nó de próximo salto, um pacote transmitido só será perdido e retransmitido se nenhum dos vizinhos selecionados o receber, o que é altamente benéfico para o ambiente sub-aquático com enlaces de baixa qualidade e altas taxas de erros. Além de uma maior taxa de entrega, o roteamento oportunístico resulta em economia de energia em redes sub-aquáticas, visto que menos pacotes precisarão ser retransmitidos, o que também reduz colisões.

Contudo, protocolos de roteamento oportunísticos que atribuem sempre o mesmo valor de prioridade para os nós vizinhos, podem reduzir o tempo de vida das redes de sensores sub-aquáticas. Nesses protocolos, a escolha do mesmo subconjunto de vizinhos, além da não rotatividade das prioridades de encaminhamento entre eles, pode levar a um gasto desigual de energia. Isso acontece porque o mesmo candidato ou alguns deles (nós centrais do ponto de vista de roteamento [Coutinho et al. 2016a]) sempre encaminharão o pacote de dados. Consequentemente, eles terão suas baterias esgotadas mais rapidamente, o que levará a existência de partições da rede, diminuindo o seu tempo de vida útil. De fato, apesar de ser um problema crítico, os protocolos de roteamento propostos até o momento vêm negligenciando esse aspecto.

Neste trabalho, é proposto um novo protocolo de roteamento oportunístico, chamado de EnOR – ***En****ergy-aware* ***O****pportunistic* ***R****outing*, para redes de sensores sub-aquáticas. O protocolo proposto tem por objetivo principal estender o tempo de vida útil da rede de sensores sub-aquática, nesse caso, definido como o tempo decorrido até que apenas uma porcentagem de nós estejam ativos. Para a seleção e priorização dos candidatos a próximo salto, EnOR considera a energia residual dos nós, o avanço do pa-

cote em direção ao destinatário e a confiabilidade do enlace para cada vizinho. A ideia básica é balancear o consumo de energia entre os candidatos por meio da rotatividade da prioridade de transmissão entre eles. Assim, o desempenho da aplicação e o tempo de vida da rede podem ser melhorados com a ausência de partições. Resultados obtidos por simulações mostram que, com relação ao tempo de vida da rede, o protocolo proposto possui desempenho superior a protocolos relacionados.

O restante deste trabalho está organizado da seguinte maneira. A Seção 2 apresenta os principais trabalhos relacionados. Em seguida, a Seção 3 apresenta brevemente a arquitetura de rede de sensores sub-aquática considerada neste trabalho, bem como o modelo matemático utilizado para estimar a probabilidade de entrega de um pacote de dados entre dois nós. A Seção 4 apresenta o protocolo proposto de roteamento oportunístico. A descrição das avaliações e os resultados obtidos por meio de simulações estão presentes na Seção 5. Finalmente, as conclusões e alguns trabalhos futuros são descritos na Seção 6.

2. Trabalhos Relacionados

Diversos protocolos de roteamento oportunísiticos foram propostos para redes de sensores sub-aquáticas. Contudo, como apresentado a seguir, não há trabalho na literatura investigando o problema do gasto desigual de energia ocasionado pela atribuição do mesmo valor de prioridade de transmissão para cada candidato, que é a estratégia usada.

Xie et al. [Xie et al. 2006] propuseram um protocolo de roteamento em que a rota e os nós intermediários para o encaminhamento do pacote é baseado na localização do nó fonte e do *sink*. No protocolo VBF (*vector-based forwarding*), cada pacote contém a localização do nó fonte, destinatário e encaminhador. O caminho para o roteamento do pacote é especificado por um vetor de roteamento (*routing vector*) do nó fonte para o destino. Os nós que recebem o pacote determinam sua posição em relação ao vetor de roteamento e, caso estejam localizados próximo ao vetor (distância menor que um limiar pré-estabelecido), incluem sua posição no pacote e encaminham o pacote. Caso contrário, o pacote é descartado. Dessa forma, os nós encaminhadores formam um "*routing pipe*" da fonte para o destino, onde os nós nesse *pipe* são candidatos a encaminharem o pacote.

A fim de se reduzir o consumo de energia em um ambiente denso em que muitos nós podem ser envolvidos no processo de encaminhamento de dados, os autores propuseram um algoritmo para o ajuste da política de encaminhamento baseado na densidade de nós, que permite aos elementos pertencentes ao *routing pipe* pesar o benefício de encaminhar o pacote e reduzir o consumo de energia via o descarte do pacote. Nesse algoritmo, quando um nó recebe um pacote, ele determina se está próximo o suficiente do *routing vector*. Se sim, o nó manterá o pacote por um período de tempo relacionado ao seu *desirableness factor*. Durante esse período, se o nó receber um pacote duplicado de outros nós, ele calculará seu *desirableness factor* relativo a esses nós e ao nó encaminhador original. Se uma restrição é satisfeita, o nó encaminhará o pacote. Caso contrário, o nó descartará o pacote.

Yan et al. [Yan et al. 2008] propuseram o primeiro protocolo de roteamento baseado na profundidade dos nós. Nesse protocolo, chamado de DBR (*Depth Based Routing Protocol*), o encaminhamento de pacotes dos nós fontes para os *sinks* na superfície ocorre exclusivamente baseado na informação da profundidade dos nós sensores. O processo de seleção do próximo salto é feito de forma gulosa. Os nós vizinhos que

estiverem em uma profundidade menor do que a do nó fonte são aptos a continuarem o encaminhamento do pacote de dados para os destinatários. O encaminhamento do pacote a partir dos nós de próximo salto é feito de forma coordenada. Isto é, os vizinhos aptos, assim que recebem o pacote, agendam a transmissão desse pacote para um tempo futuro. Esse atraso é determinado, basicamente, a partir da diferença de profundidade entre o vizinho e o nó fonte. Dessa forma, quanto mais o nó é próximo da superfície, maior será o tempo de espera até o encaminhamento do pacote. Os demais nós, ao receberem o pacote vindo, agora, de um vizinho com profundidade menor que a sua, cancelam a transmissão do pacote.

Hydrocast [Lee et al. 2010] e VAPR [Noh et al. 2013] são dois protocolos de roteamento oportunísticos em que o processo de escolha dos candidatos são também baseados na profundidade dos nós. Esses protocolos foram propostos para reduzir o alto número de pacotes redundantes, que é observado no DBR. Para isso, eles implementam uma heurística para selecionar apenas os vizinhos que estão próximos o suficiente e conseguem ouvir a transmissão do pacote uns dos outros. Assim, quando um nó encaminhador de alta prioridade transmite o pacote, os demais nós de menor prioridade conseguirão ouvir a transmissão e suprimir suas transmissões do mesmo pacote. A principal diferença entre os protocolos Hydrocast e VAPR é que o segundo usa *beacons* direcionados para determinar quais nós vizinhos não conseguem continuar o processo de entrega do pacote usando a metodologia gulosa, isto é, quais nós não estão localizados em zonas de vazio de comunicação [Coutinho et al. 2015]. O protocolo HydroCast simplesmente seleciona os candidatos a partir de um procedimento guloso, como empregado pelo DBR.

Diferentemente dos trabalhos relacionados e, mais especificamente, dos protocolos descritos acima, este trabalho propõe o protocolo de roteamento oportunístico EnOR, que visa balancear o consumo de energia dos nós encaminhadores. Para isso, o protocolo proposto considera o gasto de energia, a qualidade do enlace e o avanço do pacote a partir de cada vizinho para determinar o conjunto de candidatos (nós encaminhadores) e seus valores de prioridade de transmissão. A partir do uso dessas variáveis, a atribuição das prioridades de transmissão nos candidatos é rotacionada ao longo do tempo, o que se leva a um gasto de energia balanceado e, consequentemente, um tempo de vida prolongado da rede.

3. Preliminares

3.1. Arquitetura de Uma Rede de Sensores Sub-aquática

Neste trabalho, considera-se uma arquitetura de redes de sensores sub-aquáticas voltada para aplicações de longo prazo. Nessa arquitetura, nós sensores sub-aquáticos são presos a bóias ou âncoras e permanecem semi-estáticos em suas localizações de deposição. Arquiteturas desse tipo têm sido usadas, por exemplo, em aplicações de monitoramento de reservatórios hidrelétricos [Vieira et al. 2012]. Além dos nós sub-aquáticos, nós sorvedouros são depositados na superfície. AquaNode [O'Rourke et al. 2012] e Teledyne Benthos [Teledyne-Benthos, Acoustic Modems 2016]) são exemplos de nós sensores sub-aquáticos que podem ser usados na arquitetura de rede descrita. Por fim, é importante mencionar que, nessas arquiteturas, os nós sub-aquáticos irão periodicamente coletar dados sobre as variáveis de interesse e enviar esses dados para os nós sorvedouros existentes na superfície. Estes dados são, então, enviados para uma central de monitoramento.

3.2. O Desafio

Um dos principais desafios em redes de sensores sub-aquáticas para aplicações de longo prazo é como prolongar o tempo de vida da rede. Dado que missões marítimas para a manutenção da rede e substituição de nós e/ou baterias são extremamente custosas, é desejável manter operacional a rede de sensores sub-aquática, o maior tempo possível. Contudo, considerando as características da topologia e os protocolos de roteamento oportunísticos atuais, que não mudam as prioridades de transmissão dos nós encaminhadores, um pequeno grupo de nós centrais [Coutinho et al. 2016a], do ponto de vista de roteamento, serão muito mais demandados e, consequentemente, terão suas baterias esgotadas mais rapidamente. Isso resulta em partições da rede, o que encurta o seu tempo de vida e degrada o desempenho das aplicações.

Portanto, é necessário estudar e propor protocolos de roteamento oportunísticos, para o balanceamento no gasto de energia dos nós em redes de sensores sub-aquáticas. Nessa direção, este trabalho propõe uma heurística, visando o prolongamento do tempo de vida útil de uma rede de sensores sub-aquática. A solução proposta, em resumo, consiste no rotacionamento da prioridade de transmissão dos candidatos, ao longo do tempo, considerando diferentes variáveis. Nós candidatos que possuem valores semelhantes de qualidade do enlace, serão alternados na tarefa de encaminhamento do pacote de dados, visto que a sua prioridade de transmissão irá mudar ao longo do tempo.

4. O Protocolo de Roteamento Proposto

Esta seção descreve em detalhes o protocolo de roteamento oportunístico proposto, EnOR, que é usado para rotacionar a priodidade de transmissão dos candidatos, a fim de se alcançar um balanceamento no gasto de energia. Como a maioria dos protocolos de roteamento oportunísticos, EnOR implementa dois procedimentos principais: o procedimento de seleção dos candidatos (próximo salto) e o procedimento de atribuição de priodidades de transmissão nos candidatos. Estes procedimentos são discutidos abaixo.

Antes disso, é importante mencionar que tal como qualquer outro protocolo de roteamento geográfico, o protocolo proposto EnOR é impactado pelo problema de vazio de comunicação. Esse problema tem sido extensivamente estudado na literatura, como por exemplo em [Coutinho et al. 2015]. Este trabalho, portando, não lida com o problema de vazio de comunicação. De fato, o principal objetivo deste trabalho é focar no aspecto do gasto desbalanceado de energia, referente à tarefa de roteamento oportunístico, nos nós sensores sub-aquáticos. Esse principal objetivo do trabalho vem da necessidade de estudos e propostas de protocolos energeticamente eficientes para comunicação sem fio sub-aquática [Coutinho et al. 2016d].

4.1. EnOR: Procedimento para Seleção dos Candidatos

Em síntese, EnOR considera a confiabilidade do enlace, a energia residual e o avanço do pacote a partir de cada vizinho, a fim de selecionar o conjunto de candidatos de próximo salto, que participarão do processo de encaminhamento do pacote. Para essa seleção dos candidatos, três tarefas principais são implementadas, como descritas a seguir.

4.1.1. *Beacons* Periódicos

Usando o protocolo proposto EnOR, cada nó sensor sub-aquático transmite periodicamente um *beacon*. O pacote de *beacon* contém o identificador do nó fonte, sua localização (em termos de profundidade) e a informação sobre o seu nível atual de energia. Sempre que um nó recebe um pacote de *beacon*, ele atualiza sua tabela de vizinhos com essas informações recebidas e com o valor da distância para o nó transmissor. Cada nó sub-aquático consegue estimar sua distância para o nó que transmitiu o *beacon*, a partir da potência do sinal recebido, (RSSI, do inglês *Received Signal Strength Indicator*). Essa técnica é bem conhecida e utilizada em cenários de redes sub-aquáticas, por exemplo, por algoritmos de localização nessas redes [Han et al. 2012, Carroll et al. 2014].

4.1.2. Algoritmo de Seleção dos Candidatos

O procedimento de seleção dos candidatos propostos no EnOR é dado pelo Algoritmo 18. Seja i um nó sensor sub-aquático qualquer com um pacote de dados para transmitir. Seja N_i a tabela de vizinhos do nó i. O processo de seleção dos candidatos ao próximo salto é feito como segue. Primeiramente, o nó fonte (i) determina o valor de aptidão de cada vizinho que consegue avançar o pacote em direção ao destinatário (nós sorvedouros na superfície) (Linhas 3–8). É dito que um nó vizinho j avança o pacote se ele está localizado em uma profundidade menor do que o nó fonte, isto é, ele está mais próximo da superfície. O avanço feito pelo vizinho j é simplesmente calculado por: $P_j = \textit{Profundidade}(i) - \textit{Profundidade}(j)$.

Algoritmo 1: PROCEDIMENTO DE SELEÇÃO DOS CANDIDATOS

```
1  início
2      ▷ i: nó sensor que tem um pacote de dados para transmitir.
3      para j ∈ N_i faça
4          se P_j > 0 então
5              F.node ← j
6              F.aptidão ← aptidão (P_j, p(d_j, m), E^j_rem, E^j_init)
7          fim
8      fim
9      ordena(F.aptidão, 'descendente')
10     P_d ← 0; Γ ← ∅
11     prioridade ← 1
12     enquanto P_d < γ faça
13         Γ ← Γ ∪ {F(prioridade).node}
14         P_d ← update()
15         incrementa prioridade
16     fim
17 fim
18 retorna Γ
```

A aptidão dos nós vizinhos (Linha 6) é calculada a partir da confiabilidade do enlace do fonte (i) para o nó vizinho (j), do avanço do pacote e da energia residual do nó

vizinho (j). O valor de aptidão F_j do vizinho j é calculado por:

$$F_j = P_j \times p(d_j, m) \times \left(\frac{E^j_{rem}}{E^j_{init}} \right), \quad (1)$$

onde $P_j > 0$ corresponde ao quanto o nó j poderá avançar o pacote, em direção ao destinatário, enviado pelo nó i; $p(d_j, m)$ é a probabilidade de entrega de um pacote de m bits transmitidos pelo enlace entre os nós i e j; E^j_{rem} é o valor de energia remanescente no nó j; e E^j_{init} é o valor de energia inicial do nó j, que na verdade, é o valor inicial de todos os nós.

Os nós vizinhos aptos a continuarem a encaminhar o pacote de dados, são ordenados de acordo com seus valores de aptidão (Linha 9). Finalmente, o conjunto de nós candidatos é determinado a partir da aptidão de cada nó. Nem todos os nós aptos são considerados como candidatos. Isso porque a quantidade de candidatos é um fator que pode influenciar no desempenho da aplicação. A escolha de poucos candidatos pode resultar em uma baixa taxa de entrega, por causa da baixa confiabilidade dos enlaces acústicos. Contudo, a escolha de muitos candidatos poderá resultar em um alto atraso na entrega do pacote. No protocolo proposto, o número de candidatos é determinado pela confiabilidade agregada dos enlaces. Dessa forma, nas Linhas 10–16, os nós vizinhos aptos são escolhidos como candidatos, dos maiores valores de aptidão para as menores, até que a confiabilidade agregada dos enlaces alcance o valor desejado γ. A confiabilidade agregada é calculada a partir da confiabilidade de cada enlace, dado por:

$$P_d = 1 - \prod_{j=1}^{|\Gamma|} [1 - p(d_j, m)], \quad (2)$$

onde $p(d_j, m)$ é a probabilidade de entrega de um pacote de m bits transmitido pelo nó i para um nó j, distantes m metros.

4.1.3. Prioridade de Transmissão dos Candidatos

É importante mencionar que o EnOR intrinsicamente lida com o problema da imutabilidade das prioridades de transmissão dos candidatos. No EnOR, essas prioridades são rotacionadas ao longo do tempo. Isso acontece devido ao uso da energia residual dos candidatos, no processo de seleção deles, dado pela Eq. 1. Dessa forma, os vizinhos com baixa energia residual terão baixa prioridade de transmissão, mesmo sendo os melhores candidatos em termos de avanço do pacote em direção ao destinatário. Altas prioridades de transmissão são atribuídas a nós com maior valor de energia residual.

4.1.4. Transmissão do Pacote de Dados

Após a seleção dos candidatos e a atribuição das prioridades de transmissão, o nó fonte atual transmite o pacote em *broadcast*. No cabeçalho do pacote, o nó fonte inclui a lista de identificadores dos candidatos e a distância $D_{\max}$ para o candidato mais longe. Os identificadores dos candidatos são incluídos em formato ascendente às suas prioridades

de transmissão. Isso é para informar o candidato a sua prioridade. Dessa forma, um candidato qualquer somente encaminha o pacote se ele não detectar o encaminhamento do mesmo pacote a partir de um nó com prioridade mais alta. A distância entre o nó fonte e o candidato mais longe é usada para a coordenação das transmissões por meio de temporizadores, como será descrito na próxima seção.

4.2. EnOR: Algoritmo de Coordenação das Transmissões

A coordenação das transmissões dos candidatos é um dos problemas mais desafiadores no projeto de protocolos de roteamento oportunísticos. Uma discussão detalhada sobre esse aspecto é apresentada em [Coutinho et al. 2016c].

EnOR implementa um procedimento baseado em temporizador, para a coordenação das transmissões dos candidatos. Nessa abordagem, um *slot* para transmissão é atribuído a cada candidato, de acordo as suas prioridades. Isso pode ser implementado localmente, sem o auxílio de pacotes de controle, o que é desejado para redes de sensores sub-aquáticas. O procedimento adotado pelo EnOR funciona como segue. Cada nó candidato agenda a transmissão do pacote recebido para um tempo futuro. Esse atraso é proporcional ao nível de prioridade do candidato. Quando esse tempo expira, o candidato, então, encaminha o pacote de dados, caso ele não detecte que houve o encaminhamento do mesmo pacote, por um nó de maior prioridade. Essa abordagem é simples e escalável. Além disso, ela não acarreta em *overhead* para a rede. Contudo, uma vez que nós candidatos não encaminham o pacote imediatamente, tem-se um maior atraso para a entrega do pacote.

4.2.1. Tempo de Espera Antes do Encaminhamento

Em uma coordenação baseada em temporizador, um nó deverá esperar um tempo relativo à sua prioridade. Esse tempo deve ser um pouco maior ao tempo necessário pelo seu predecessor de alta prioridade para receber, processar e encaminhar o pacote. Além disso, esse atraso deverá considerar o tempo de propagação do pacote encaminhado pelo nó de maior prioridade. Somente após esse tempo, caso o candidato em questão não detecte o encaminhamento do pacote por um nó de maior prioridade, ele encaminha o pacote. Para determinar esse tempo de retenção do pacote, o procedimento de seleção do conjunto de candidatos deve garantir que todos os nós do conjunto são vizinhos, que permitem detectarem as transmissões dos outros e que conheçam as distâncias entre si. A informação da distância entre eles é incluída no cabeçalho do pacote de dados. Para evitar um longo cabeçalho no pacote do EnOR, o tempo de espera de pacote, com base na prioridade dos nós, deve considerar a maior distância entre o remetente e os candidatos. Assim, ao receber um pacote de dados, o nó candidato retém o pacote durante um certo tempo T_h dado como:

$$T_h(p) = \begin{cases} \dfrac{R - D_{\max}}{v}, \text{ if } p = 1 \\ \dfrac{R + p \times D_{\max}}{v}, \text{ if } p > 1, \end{cases} \tag{3}$$

onde R é o valor do raio de comunicação, $D_{\max}$ é a distância entre o nó fonte e o candidato mais longe, p é a prioridade do candidato que recebeu o pacote e está calculando seu

tempo de retenção do pacote e v é a velocidade de propagação do sinal acústico, dado aproximadamente por 1500 m/s.

4.2.2. Supressão de Transmissões Redundantes

A supressão de transmissões desnecessárias é importante, visto que encaminhamentos redundantes de pacotes afetam diretamente o desempenho da rede. Nos protocolos VBF [Xie et al. 2006] e DBR [Yan et al. 2008], um nó de baixa prioridade cancelará a transmissão de um pacote agendado se o nó escutar a transmissão do mesmo pacote por um nó de alta prioridade. Em ambos os protocolos, espera-se ter um número elevado de pacotes redundantes devido ao problema do terminal oculto.

A fim de se reduzir as transmissões redundantes ocasionadas pelo problema do terminal oculto, os protocolos Hydrocast [Lee et al. 2010], VAPR [Noh et al. 2013] e GEDAR [Coutinho et al. 2016b] procuram selecionar como candidatos, os nós vizinhos que estão distantes entre si e no máximo o valor do rádio de comunicação. Assim, um nó candidato pode ouvir as transmissões de seus vizinhos. No entanto, isso não é suficiente para evitar o problema de terminal oculto dada a atenuação do sinal e o ambiente sub-aquático ruidoso.

EnOR emprega um mecanismo de supressão ativo. No mecanismo proposto, sempre que um nó sub-aquático de alta prioridade encaminha o pacote de dados, todos os nós de baixa prioridade que ouvem a transmissão cancelam o encaminhamento do mesmo pacote. Além disso, o nó fonte, ao receber seu pacote encaminhado por um candidato, envia um curto pacote de supressão, contendo o seu identificador e o número de sequência do pacote transmitido. Assim, todos os candidatos que recebem esse pacote de supressão, cancelam o encaminhamento do pacote de dados referente a esse alerta.

5. Avaliação de Desempenho

Nesta seção, é avaliado o desempenho do protocolo de roteamento oportunístico EnOR. Seu desempenho é comparado com outros dois protocolos de roteamento oportunísticos para redes sub-aquáticas: DBR [Yan et al. 2008] e VAPR [Noh et al. 2013].

5.1. Metodologia

A avaliação de desempenho realizada neste trabalho foi feita através de simulações, usando o simulador Aqua-Sim v.1.0 [Xie et al. 2009]. Aqua-Sim é um simulador de rede sub-aquática, baseado no ns-2.30. Esse simulador implementa as deficiências da comunicação acústica, tais como a atenuação do sinal acústico e as colisões de pacotes no ambiente sub-aquático.

Com relação à topologia de rede, nós sensores sub-aquáticos são distribuídos aleatoriamente na região de interesse. Para a deposição dos sorvedouros, a área de interesse é dividida em uma grade de quatro quadrados de lado l igual a 500 m. Em cada quadrado, 16 sorvedouros são distribuídos aleatoriamente. Em termos de densidade da rede, dois cenários são considerados: 150 e 350 nós sensores sub-aquáticos. Nas simulações, as configurações dos parâmetros são baseadas nos trabalhos encontrados na literatura, tais como em [Yan et al. 2008, Lee et al. 2010, Noh et al. 2013]. Esses valores

Table 1. Parâmetros de simulação e propriedades da topologia da rede.

Parâmetro	Valor	Parâmetro	Valor
Área de interesse	$(1000\,\text{m})^3$	Raio de comunicação (r_c)	250 m
Número de sub-áreas (K)	4	Largura de banda (B)	20 kb/s
Tamanho da sub-área (l)	500 m×500 m	Protocolo MAC	CSMA
Número de *sinks* ($\mathcal{N}_s/K$)	16 *sinks*/sub-área	Tam. do pacote de dados	100 bytes
Quant. de nós sensores	150, 350 nodes	Custo energético para transmissão (P_t)	4 W
Duração da simulação	2 dias	Custo energético para recepção (P_r)	0.65 W
Taxa de geração de pacotes (λ)	0.01 pac./min	Limiar de qualidade de link (γ)	0.9

são mostrados na Tabela 1. Os resultados apresentados correspondem a um valor médio de 50 execuções com intervalo de confiança de 95 %.

EnOR usa a confiabilidade do enlace para a seleção do conjunto de candidatos. Neste trabalho, utilizamos o modelo de canal acústico de Urick [Urick 1983], para estimar a confiabilidade do enlace, em termos de probabilidade de entrega de pacotes, entre dois nós sensores sub-aquáticos, como descrito a seguir.

A perda na potência do sinal em caminho de distância *d* em uma frequência *f* devido ao desvanecimento em larga escala é dada por:

$$A(d, f) = d^k a(f)^d. \tag{4}$$

Na Eq. 4, k é o fator de dispersão e $a(f)$ é o coeficiente de absorção. A geometria da propagação é descrita usando um fator de dispersão ($1 \leq k \leq 2$)ç para um cenário prático, k é dado por 1.5. O coeficiente de absorção $a(f)$ é calculado pela fórmula de Thorp [Brekhovskikh and Lysanov 2003], dado por:

$$10 \log a(f) = \frac{0.11 \times f^2}{1 + f^2} + \frac{44 \times f^2}{4100 + f} + 2.75 \times 10^{-4} f^2 + 0.003. \tag{5}$$

A média da razão entre sinal-ruído (SNR – *Singal-to-Noise Ratio*) por uma distância *d* é dada por:

$$\Gamma(d) = \frac{E_b/A(d, f)}{N_0} = \frac{E_b}{N_0 d^k a(f)^d}, \tag{6}$$

onde E_b e N_0 são constantes que representam a energia média de transmissão por bit e a densidade de ruído potência em um canal sem desvanecimento e com ruído gaussiano branco aditivo (AWGN – *non-fading additive white Gaussian noise*). Como mostrado em [Stojanovic 1996] e [Carbonelli and Mitra 2006], o modelo de desvanecimento Rayleigh é usado para modelar desvanecimento em pequena escala, onde o SNR tem a seguinte distribuição de probabilidade:

$$p_d(X) = \int_0^\infty \frac{1}{\Gamma(d)} e^{-\frac{X}{\Gamma(d)}}. \tag{7}$$

A probabilidade de erro pode ser avaliada como:

$$p_e(d) = \int_0^\infty p_e(X) p_d(X) dX. \tag{8}$$

onde $p_e(X)$ é a probabilidade de erro para uma modulação arbitrária para valores específicos do SNR X. Neste trabalho, usamos a modulação BPSK (*Binary Phase Shift Keying*) que é vastamente utilizada no estado da arte dos modems acústicos [Freitag et al. 2005, Stojanovic and Preisig 2009, Yang et al. 2009]. Na modulação BPSK, cada símbolo carrega um bit. A probabilidade de erro no bit, dada uma distância d, é dada por [Rappaport 2002]:

$$p_e(d) = \frac{1}{2}\left(1 - \sqrt{\frac{\Gamma(d)}{1+\Gamma(d)}}\right). \tag{9}$$

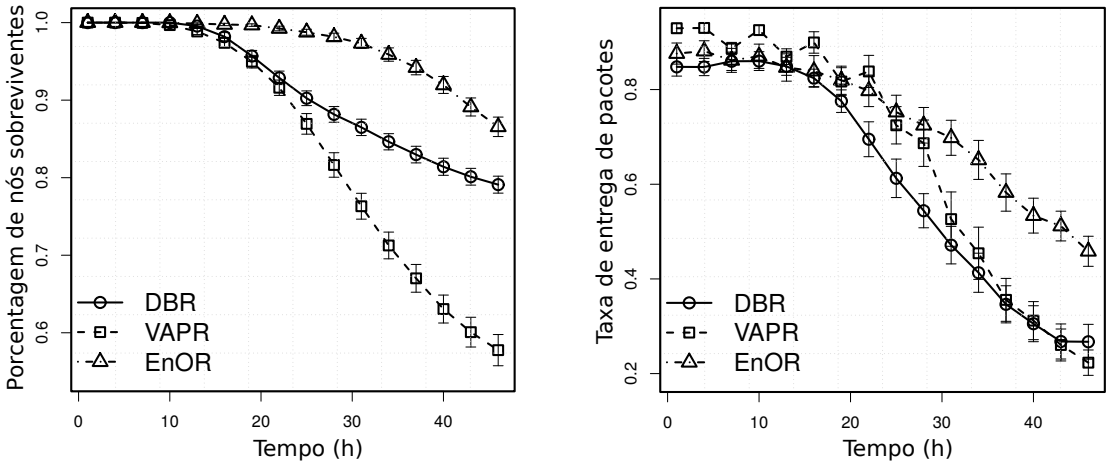
Assim, a probabilidade de entrega de um pacote com tamanho m bits, transmitido entre um par de nós com distância de d metros, pode ser calculada por:

$$p(d, m) = (1 - p_e(d))^m. \tag{10}$$

5.2. Resultados

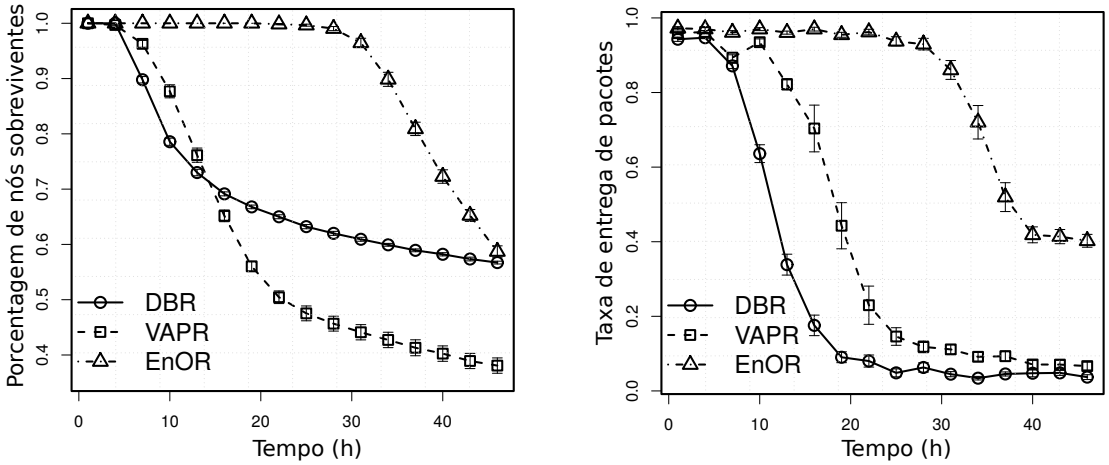
As Figuras 1(a) e 2(a) mostram a taxa de nós sobreviventes ao longo do tempo. Nota-se que a fração de nós sobreviventes reduz rapidamente no cenário de rede de alta densidade, mostrado na Figura 2(a). Isso acontece devido ao custo de energia para recepção de pacotes e o meio *broadcast* de transmissão sem fio, que faz com que todos os nós no alcance de comunicação do nó fonte recebam o pacote de dados, mesmo quando este é endereçado a um nó específico. Os resultados mostram que o protocolo proposto estende significativamente a vida útil da rede, quando comparado com os protocolos DBR e VAPR. Isto acontece devido ao rotacionamento das prioridades de transmissão dos candidatos, em função de sua energia residual. Com relação aos protocolos DBR e VAPR, os resultados mostram que a exaustão das baterias dos nós ocorre de forma não homogênea em ambos os cenários, devido ao fato de que esses protocolos selecionam os candidatos apenas considerando o avanço do pacote a partir de cada vizinho. Isso faz com que o mesmo candidato seja o nó de maior prioridade e encaminhador do pacote.

O comportamento acima mencionado reflete no desempenho da rede. As Figuras 1(b) e 2(b) mostram a taxa de entrega de pacotes. Essa métrica é calculada como a porcentagem dos pacotes de dados recebidos durante um período de tempo pré-determinado (1 hora). A taxa de entrega de dados dos protocolos de roteamento oportunísticos avaliados, diminuem em função do tempo. Essa diminuição é associado ao fato que alguns nós sensores esgotam suas baterias ao longo do tempo, como corroborado pelas Figuras 1(a) e 2(a). No EnOR, a taxa de entrega de pacotes é maior que 90% em 87% do tempo para cenários de baixa densidade (veja Figura 1(b)) em comparação com VAPR e DBR, respectivamente. Isso porque mais nós estão ativos, como observado para cenários de alta densidade como mostrado na Figura 2(a).



(a) Fração de nós sobreviventes (b) Taxa de entrega de pacotes

Figure 1. Resultados: Cenário com 150 nós sensores sub-aquáticos



(a) Fração de nós sobreviventes (b) Taxa de entrega de pacotes

Figure 2. Resultados: Cenário com 350 nós

6. Conclusão

Neste trabalho, abordamos o problema de imutabilidade da prioridade de transmissão de nós candidatos, em protocolos de roteamento oportunísticos em redes de sensores sub-aquáticas. Propusemos um novo protocolo de roteamento oportunístico, chamado EnOR, que considera a energia residual dos vizinhos, a confiabilidade do enlace para cada vizinho e o quanto cada vizinho avança o pacote de dados em direção ao destinatário. No protocolo proposto, essas métricas foram utilizadas para selecionar os nós candidatos e priorizar seus encaminhamentos de pacotes de dados. O principal objetivo da nossa proposta foi alcançar um balanceamento de energia nos nós sensores, para estender o tempo de vida da rede. Os resultados de simulação confirmaram que o protocolo proposto efetivamente prolonga a vida útil dos nós sensores sub-aquáticos. Como consequência imediata, o desempenho da rede foi melhorado. Como trabalho futuro, pretendemos combinar o EnOR com a solução para o problema de vazio de comunicação [Coutinho et al. 2016d].

References

Akyildiz, I. F., Pompili, D., and Melodia, T. (2005). Underwater acoustic sensor networks: research challenges. *Ad Hoc Networks*, 3(3):257–279.

Brekhovskikh, L. M. and Lysanov, Y. (2003). *Fundamentals of Ocean Acoustics*. Springer.

Carbonelli, C. and Mitra, U. (2006). Cooperative multihop communication for underwater acoustic networks. In *WUWNet'06*, pages 97–100.

Carroll, P., Mahmood, K., Zhou, S., Zhou, H., Xu, X., and Cui, J.-H. (2014). On-demand asynchronous localization for underwater sensor networks. *IEEE Trans. on Signal Processing*, 62(13):3337–3348.

Coutinho, R. W., Boukerche, A., Vieira, L. F., and Loureiro, A. A. (2016a). A novel centrality metric for topology control in underwater sensor networks. In *Proc. of the 19th ACM MSWiM*.

Coutinho, R. W. L., Boukerche, A., M.Vieira, L. F., and Loureiro, A. A. F. (2016b). Geographic and opportunistic routing for underwater sensor networks. *IEEE Trans. on Computers*, 65(2):548–561.

Coutinho, R. W. L., Boukerche, A., Vieira, L. F. M., and Loureiro, A. A. (2015). A novel void node recovery paradigm for long-term underwater sensor networks. *Ad Hoc Networks*, 34:144 – 156.

Coutinho, R. W. L., Boukerche, A., Vieira, L. F. M., and Loureiro, A. A. F. (2016c). Design guidelines for opportunistic routing in underwater networks. *IEEE Commun. Magazine*, 54(2):40–48.

Coutinho, R. W. L., Boukerche, A., Vieira, L. F. M., and Loureiro, A. A. F. (2016d). On the design of green protocols for underwater sensor networks. *IEEE Commun. Magazine*, 54(10):67–73.

Freitag, L., Grund, M., Singh, S., Partan, J., Koski, P., and Ball, K. (2005). The whoi micro-modem: An acoustic communcations and navigation system for multiple platforms. In *Oceans'05*.

Han, G., Jiang, J., Shu, L., Xu, Y., and Wang, F. (2012). Localization algorithms of underwater wireless sensor networks: A survey. *Sensors*, 12(2):2026–2061.

Lee, U. et al. (2010). Pressure routing for underwater sensor networks. In *Proc. of the IEEE INFOCOM*, pages 1–9.

Noh, Y., Lee, U., Wang, P., Choi, B. S. C., and Gerla, M. (2013). VAPR: void-aware pressure routing for underwater sensor networks. *IEEE Trans. on Mobile Comput.*, 12(5):895–908.

O'Rourke, M., Basha, E., and Detweiler, C. (2012). Multi-modal communications in underwater sensor networks using depth adjustment. In *Proc. of the 17th ACM WUWNet*, pages 31:1–31:5.

Partam, J., Kurose, J., and N., L. B. (2006). A Survey of Practical Issues in Underwater Networks. In *Proc. of the ACM WUWNet*.

Rappaport, T. (2002). *Wireless Communications: Principles and Practice*. Prentice Hall.

Stojanovic, M. (1996). Recent Advances in High-speed Underwater Acoustic Communications. *IEEE Journal of Oceanic Engin.*, pages 125–136.

Stojanovic, M. and Preisig, J. (2009). Underwater acoustic communication channels: Propagation models and statistical characterization. *IEEE Communications Magazine*, 47(1):84–89.

Teledyne-Benthos, Acoustic Modems (2016). `http://teledynebenthos.com`. [Online].

Urick, J. R. (1983). *Principles of Underwater Sound*. McGraw-Hill.

Vieira, L. F. M., Vieira, M. A. M., Pinto, D., Nacif, J. A. M., Viana, S. S., and Vieira, A. B. (2012). Hydronode: An underwater sensor node prototype for monitoring hydroelectric reservoirs. In *Proc. of the 17th ACM WUWNet*, pages 43:1–43:2.

Xie, P., Cui, J.-H., and Lao, L. (2006). VBF: Vector-Based Forwarding Protocol for Underwater Sensor Networks. In *IFIP Networking'06*.

Xie, P., Zhou, Z., Peng, Z., Yan, H., Hu, T., Cui, J.-H., Shi, Z., Fei, Y., and Zhou, S. (2009). Aqua-Sim: An NS-2 based simulator for underwater sensor networks. In *Proc. of the MTS/IEEE OCEANS*, pages 1–7.

Yan, H., Shi, Z., and Cui, J.-H. (2008). DBR: depth-based routing for underwater sensor networks. In *Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, volume 4982 of *Lecture Notes in Computer Science*, pages 72–86. Springer Berlin Heidelberg.

Yang, H., Liu, B., Ren, F., Wen, H., and Lin, C. (2009). Optimization of energy efficient transmission in underwater sensor networks. In *Proc. of the IEEE GLOBECOM*, pages 1–6.

Parallel and Efficient IP Lookup using Bloom Filters on Intel® Xeon Phi™

Alexandre Lucchesi[1], **André C. Drummond**[1], **George Teodoro**[1]

[1]Department of Computer Science
University of Brasília
Brasília, Brazil

lucchesi@aluno.unb.br, {andred, teodoro}@unb.br

Abstract. *The IP lookup phase is the core operation in packet forwarding, which is implemented via a Longest Prefix Matching (LPM) to find the next hop for every input address. In this work, we evaluate the use of parallel techniques to develop a highly optimized IP lookup algorithm that employs Bloom filters and hash tables. More specifically, we investigate the implementation of our algorithm on multi-core CPUs and on the Intel® Xeon Phi™ (Intel Phi) many-core coprocessor. Our analysis includes the efficient parallelization of our Bloom filters algorithm on both devices, and the experimental results show that we were able to attain high performance with this solution (over 88 million lookups per second on a single Intel Phi for IPv6). We also compared the Bloom filters optimized solution to an efficient approach based on the Multi-Index Hybrid Trie (MIHT). This comparison shows that the most efficient sequential algorithm may not be the best option in a parallel setting. Instead, it is necessary to evaluate the processors characteristics, algorithms compute/data demands and data structures employed to analyze how the algorithms will benefit from the target computing device. These findings are also important to new efforts in algorithmic developments in the topic, which have been highly focused on sequential solutions.*

1. Introduction

The use of software routers is motivated due to its easy extensivity, programmability, and good cost-benefit. However, these routers are required to attain high packet forwarding rates, which is a challenging task that may be reached with more efficient algorithms and/or with the use of high-performance parallel computing techniques. The calculation of the next hop for input packets is a core operation in the forwarding phase of routers. Since the development of CIDR, routers are required to run a Longest Prefix Matching (LPM) algorithm in order to determine the next hop for each received packet. This task is commonly referred to as IP lookup and is often the performance bottleneck in high-performance software routers.

In this work, we investigate the Intel Phi coprocessor as a platform for the implementation of efficient LPM algorithms for IP lookup. The Intel Phi is a highly parallel platform that supports up to 244 threads, provides 512-bit SIMD instructions, and has a large memory bandwidth. These characteristics make the Intel Phi an attractive platform for the implementation of LPM for multiple incoming IP addresses. We have designed a parallel algorithm that uses Bloom filters and hash tables to efficiently find the LPM for both IPv4 and IPv6. The implementation leverages the Intel Phi capabilities to mitigate

the main drawbacks of the algorithm — namely, the high costs to compute hashes during lookup/store operations and the high memory bandwidth requirements. This is achieved with the use of vectorization to reduce the costs of hashing and thread-level parallelism to increase concurrency and throughput.

In order to evaluate our algorithm, we have compared the implementation to a parallel version of the Multi-Index Hybrid Trie (MIHT). The MIHT is a state-of-the-art sequential lookup algorithm that has been shown to attain better performance than well-known tree/trie based algorithms: the Binary Trie, the Prefix Tree, the Priority Trie, the DTBM, the 4-MPT and the 4-PCMST [Lin et al., 2014]. The experimental evaluation of the algorithms have shown that our optimized Bloom filters algorithm was able to outperform the MIHT approach in both sequential and parallel executions on the Intel Phi. In a parallel execution using IPv4 and IPv6 prefix datasets our Bloom filters optimized algorithm was, respectively, up to 4.31× and 5.39× faster than MIHT. The results show that, although the MIHT is a very memory-efficient algorithm, it presents fewer opportunities for optimizations and worse scalability on Intel Phi. For instance, the use of vector SIMD instructions available in most of the modern device architectures can be leveraged to improve the Bloom filters approach, but is not effective for MIHT because of the irregular nature of the data structures it uses. The main contributions of this paper can be summarized as:

- We implement an efficient version of the Bloom filters based LPM algorithm that fully exploits the Intel Phi capabilities.
- We evaluate the compromises of using the Intel Phi coprocessor and modern CPUs to the IP lookup problem using two algorithmic strategies.
- We propose a novel approach combining dynamic programming and Controlled Prefix Expansion [Venkatachary and Varghese, 1998] (DPCPE) to enhance the performance of IPv6 lookups in the Bloom filters based algorithm.
- We show in our experimental evaluation that the most efficient sequential algorithm may not be the better solution in a parallel setting. Instead, the possibility of adapting the algorithm to fully utilize the processor features and parallelism can lead to higher performance.

The rest of this paper is organized as follows. Section 2 describes the Intel Phi coprocessor and related works. Section 3 describes the use of Bloom filters to solve the IP lookup problem. Section 4 details the algorithm design, including the optimizations, parallelization strategies, and relevant implementation details. We experimentally evaluate our algorithm in Section 5 and present conclusions and future directions in Section 6.

2. Background and Related Work

2.1. Intel® Xeon Phi™

The Intel Phi coprocessor is based on the Intel Many Integrated Core architecture (MIC), which consists of many simplified, power efficient, and in-order computing cores equipped with a 512-bit vector processing unit (SIMD unit). The Intel Phi 7120P used in this work has 61 cores with a four-way hyperthreading that leads to the simultaneous execution of up to 244 threads, and a theoretical peek performance of 1.2 teraflops. Each core is clocked at 1.333 GHz, has 512 KB private L1 cache, and a L2 cache of about 30 MB (aggregate for all cores) that is kept fully coherent among cores via a directory tag mechanism. The computing cores, caches, and memory controllers are connected through a

high bandwidth bidirectional ring interconnect (352 GB/s). The Intel Phi is also equipped with 16 GB of GDDR5 of main memory. The combination of the hyperthreading and the high-bandwidth memory is efficient for hiding memory accesses latency that is important for memory-intensive applications.

The MIC architecture combines features of general-purpose CPUs and many-core processors or accelerators to provide an easy to program and high-performance computing environment [Jeffers and Reinders, 2013]. It is based on a x86 instruction set and supports traditional parallel and communication programming models, such as OpenMP (Open Multi-Processing), Pthreads (POSIX Threads Programming), MPI (Message Passing Interface), etc. It runs applications in *native mode* or *offload mode*. In native mode, the user directly accesses the coprocessor via SSH connections to run the application within it. This is possible because Phi runs a simplified operating system. In offload mode, the programmer selects parts of the application, usually annotating the source code with specific pragmas, to be automatically transferred and executed on the coprocessor.

2.2. Previous work

The trie/tree-based is a popular class of IP lookup algorithms [Ruiz-Sanchez et al., 2001], and finding the LPM in these algorithms usually consists of sequentially traversing a sequence of nodes. Therefore, these algorithms strive to reduce the number of required memory accesses as a means to speed up the lookup process. For instance, in order to achieve that, the Multi-Index Hybrid Trie (MIHT) [Lin et al., 2014] employs space-efficient data structures, such as B^+ trees and Priority Tries [Lim et al., 2010]. Recently, the use of compressed trie data structures has also been proposed [Rétvári et al., 2013, Asai and Ohara, 2015]. Nevertheless, trie/tree-based schemes commonly share the characteristic of being memory-intensive. Another interesting class of algorithms for IP lookup is based on Bloom filters [Dharmapurikar et al., 2006, Lim et al., 2014]. These algorithms are compute-intensive and may require many hash calculations during each lookup/store operation. Hashing is used within the Bloom filters as a means to avoid unnecessary memory accesses to hash tables (where the data is actually stored).

A wide range of hardware architectures has been used to implement IP lookup algorithms, including CPU, FPGA, GPU and many-cores [Yang et al., 2015]. In [Ni et al., 2015], a Parallel Bloom Filter (PBF) was implemented in the Intel Phi coprocessor. PBF was proposed to reduce synchronization overhead and improve cache locality in many-core platforms. In this work, we also implement an algorithm that uses Bloom filters in the Intel Phi specialized for IP lookup and, as such, our approach is different both in the algorithmic and implementation levels. First, PBF uses locks in the Bloom filters to avoid data races in concurrent update/lookup operations and enforces sequential consistency by reordering the responses after processing the requests in parallel. Our approach, in the other hand, is built on top of the ideas of [Dharmapurikar et al., 2006] and includes several optimizations targeting its efficient execution on the Intel Phi for IP lookup. Therefore, we have designed a Bloom filters based LPM algorithm specifically optimized for performing the IP lookup task for both IPv4 and IPv6. As will be presented in the experimental evaluation, these optimizations are crucial to attain high performance.

3. Bloom Filters for IP Lookup

The use of Bloom filters coupled with hash tables for computing IP Lookups has been proposed in [Dharmapurikar et al., 2006]. The standard algorithm has been developed

using 32 pairs of Bloom filters and hash tables for IPv4 lookups, whereas this number would increase to 64 in the case of IPv6. For the sake of simplicity, in the remaining of this section, we describe the algorithm in the context of IPv4, while modification for its efficient execution with IPv6 are presented in the next section.

A Bloom filter is an efficient data structure for membership queries with tunable false positive errors [Bloom, 1970] widely used for web caching, intrusion detection, content based routing and LPM [Dharmapurikar et al., 2006]. In essence, a Bloom filter consists of a bit-vector used to represent a set of values. A Bloom filter is programmed by computing hash functions on each element it stores, and by setting the corresponding indices in the bit-vector. Further, to check if a particular value is in the set, the same hash functions are computed on the input value and bits in the bit-vector structure addressed by the hash values are verified. The value is said to be contained in the set with a given probability only if all bits are set. The standard Bloom filter does not support removal of elements from the set, because it does not control the case in which a bit is set multiple times due to the insertion of different values whose hashes collide. This is addressed with the use of Counting Bloom Filters [Fan et al., 2000], which associates a counter with each entry of the bit-vector to store the number of times a given bit-vector entry was set or unset. We have chosen to implement this approach because it provides a fair comparison with other algorithms that also feature dynamic forwarding tables.

The lookup operations in the standard Bloom filters algorithm are executed within multiple sets of filters and hash tables — one for each possible IP prefix length. As network addresses in IPv4 are 32-bit long, they require the algorithm to employ 32 Bloom filters with their respective 32 hash tables. Each hash table stores their corresponding [prefix, next hop] pairs and any other relevant routing information, such as metric, interface, etc. If a default route exists, it is stored in a separate field in the forwarding table data structure. Let $F = \{(f_1, t_1), (f_2, t_2), \ldots, (f_{32}, t_{32})\}$ be the set of Bloom filters (f_i) and associated hash tables (t_i) that form an IPv4 forwarding table, where (f_1, t_1) corresponds to the data structures that store 1-bit long prefixes, (f_2, t_2) corresponds to the data structures that store 2-bit long prefixes, and so on. In addition, let $len(f_i)$ be the length of the bit-vector of the i-th Bloom filter, where $1 \leq i \leq 32$. The forwarding table construction is as follows. For every network prefix p of length l to be stored, k hash functions are computed, yielding k hash values: $H = \{h_1, h_2, \ldots h_k\}$. The algorithm uses H to set the k bits corresponding to the indices $I = \{h_i \bmod len(f_l) \mid 1 \leq i \leq 32\}$ in the bit-vector of the Bloom filter f_l. It also increments the corresponding counters in the array of counters of f_l.

The lookup process is similar to the insertion. Given an input destination address DA, the algorithm first extracts its segments or prefixes. Let $S_{DA} = \{s_1, s_2, \ldots, s_{32}\}$ be the set of all the segments of a particular address DA, where s_i is the segment corresponding to the first $1 \leq i \leq 32$ bits of DA. For each $s_i \in S_{DA}$, k hash functions are computed, yielding k hash values for each segment: $H = \{(h_1, h_2, \ldots h_k)_1, (h_1, h_2, \ldots h_k)_2, \ldots, (h_1, h_2, \ldots h_k)_{32}\}$. The element $H'_i \in H$ is used to query the Bloom filter $f_i \in F$. The process is as follows: the algorithm checks the k bits in the bit-vector of f_i using the indices $I = \{h_j \bmod len(f_i) \mid h_j \in H'_i, 1 \leq j \leq k \text{ and } 1 \leq i \leq 32\}$. The result of this process is a *match vector* $M = \{m_1, m_2, \ldots, m_{32}\}$ containing the answers of each Bloom filter, i.e., each $m_i \in M$ indicates whether a match occurred or not in f_i. The match vector M is used to query the associated hash tables. The

search begins by sequentially performing queries to the associated hash tables by traversing M backwards, i.e., starting in m_{32}. This is because we are interested in the LPM. If the algorithm finds the next hop (a true match) for a given DA in the pair (f_i, t_i), it is the LPM. As Bloom filters may produce false-positives but never false negatives, when a filter does not match a segment, i.e., $m_i \in M$ indicates a mismatch, the algorithm can safely skip to the next Bloom filter f_{i-1} (if $i \geq 2$) without touching its associated hash table t_i. This process continues until the LPM is found or all pairs (f_i, t_i) are unsuccessfully searched. Please, note that false-positives will only lead to extra hash table searches, and the actual result of the algorithm will be the same regardless of that ratio.

4. Bloom Filters Optimizations and Parallelization

This section describes the optimizations implemented in the standard Bloom filters IP lookup algorithm, as well as its parallelization targeting the Intel Phi. The CPU parallel versions employed similar parallelization strategies, but differ with respect to the instruction-level parallelism that used auto-vectorization. The baseline implementation on which our work is built incorporates the following optimizations: the use of an array of counters to allow FIB updates, asymmetric memory allocation proposed in [Dharmapurikar et al., 2006], and Controlled Prefix Expansion (CPE) [Venkatachary and Varghese, 1998] to reduce the number of required data structures.

4.1. Optimizing the Hash Calculations

Hashing quality is a central performance aspect of the algorithm because it is closely related to the efficiency of Bloom filters and hash tables. In the Bloom filters data structure, it affects both the false positive ratio (FPR) and the amount of memory required. With respect to the associated hash tables, the better the quality of the hash, less collisions are likely to happen and, as a consequence, the lookup process will also be faster.

In order to improve the algorithm performance we have (i) accelerated the hash calculations with the use of instruction-level parallelism or vectorization, as discussed in detail in Section 4.4; (ii) reduced the cost of hashing by combining the output of two hash calculations to generate more hashes; and, (iii) implemented and evaluated the reuse of hash values to minimize the overall hash calculations. The reuse affects both the lookup and update operations. The generation of extra hashes was performed through the use of a well-known technique that consists of using a simple linear combination of the output of two hash functions $h_1(x)$ and $h_2(x)$ to derive additional hash functions in the form $g_i(x) = h_1(x) + i \times h_2(x)$. This technique results in much faster hash calculations and can be effectively applied in the Bloom filters and related data structures, such as hash tables, without affecting the asymptotic false positive probabilities [Kirsch and Mitzenmacher, 2008]. We have also proposed the reuse of one of the hashes calculated to search or store a key in the Bloom filter to address its associated hash table. This avoids the calculation of another hash whenever a hash table is visited.

4.2. A New Dynamic Programming CPE (DPCPE) for Efficient IPv6 Lookup

Another crucial optimization we have implemented for IPv6 is the use of CPE to reduce the number of required sets of Bloom filters and hash tables in the algorithm. This technique consists of expanding every prefix of a shorter length to multiple, equivalent, prefixes of a greater length, so that the number of distinct prefix lengths and, consequently,

filters and hash tables, is reduced. In IPv4, we used CPE to expand prefixes into two groups: $G_1 \in [21–24]$ and $G_2 \in [25–32]$. After the CPE, G_1 has only 24-bit prefixes and G_2 has only 32-bit prefixes, and two sets of Bloom filters and hash tables are allocated to store these prefixes. A Direct Lookup Array (DLA) is allocated to store the next hops of the remaining prefixes, whose lengths are ≤ 20 bits, using the prefixes themselves as the indices. Except for the DLA, the lookup process is identical to the standard Bloom filters (discussed in Section 3). The algorithm sequentially searches the two sets of Bloom filters and hash tables (starting from G_2, since it stores the longest prefixes) and, if the LPM is not found, the next hop stored in the DLA position indexed by the first 20 bits of the input address is returned (it may be the default route). This way we are able to bound the worst-case lookup scenario to two queries (G_1, G_2) and one memory access (DLA), as detailed in [Dharmapurikar et al., 2006]. Note that the trade-off of CPE is faster search on the cost of increased memory footprint, as shown in Table 2.

The previous work on IP Lookup using Bloom filters [Dharmapurikar et al., 2006] has reported that this technique would not be efficient for IPv6. This is because of the "strides" between hierarchical boundaries of IPv6 addresses, which would result in a very high use of memory after expansion. However, we have proposed and implemented an algorithm based on dynamic programming (DPCPE) that groups prefix lengths and performs the expansion with a limited additional memory demand.

The DPCPE algorithm works as follows. Let $L = \{l_1, l_2, \ldots, l_{64}\}$ be the prefix distribution of a IPv6 forwarding table, where l_i is the number of unique prefixes of length i (in bits). Given a desired number of expansion levels n, the algorithm uses dynamic programming to compute the set of lengths to be used in order to minimize the total number of prefixes in the resulting forwarding table. DPCPE always starts by picking the length 64, since it is the largest prefix length for IPv6 and, as such, its inclusion is required for correctness (i.e. every IPv6 prefix can, *theoretically*, be expanded to one or more 64-bit prefixes). Let $S = \{64\}$ represent the initial set of resulting lengths and $C = \{1, 2, \ldots, 63\}$ represent the initial set of candidate lengths. While $|S| < n$, the algorithm iteratively removes an element $l \in C$ and inserts it into S. In each iteration, the length l is selected by mapping a cost function f over all possible sets of lengths and choosing the length associated with the smaller cost. For instance, in the second iteration (assuming $n \geq 2$), f is mapped over the set $Q = \{\{l, 64\} \mid l \in C\}$ and the value l from the set that resulted in the minimum cost is selected. The cost function f takes as input L and a set of expansion levels $Q' \in Q$. It then computes the resulting number of prefixes after expanding L to Q'. The (maximum) number of prefixes, resulting from expanding a prefix of length $l_i \in L$ to $q \in Q'$ (such that, $i < q$), is defined as $2^{q-i} \times l_i$. Note that f does not take into account the problem of prefix capture [Venkatachary and Varghese, 1998], which happens whenever a prefix is expanded to one or more existing prefixes in the database. In this case, the existing longer prefix "captures" the expanded one, which is ignored. Therefore, although DPCPE is not guaranteed to return the optimal solution, it usually returns solutions that work better in practice for the Bloom filters algorithm than directly using the database with no preprocessing, as presented in Section 5.5.

4.3. Thread-Level Parallelism (TLP)

Due to its regular data structures, the Bloom filters algorithm exposes multiple opportunities for parallelism. For instance, in [Dharmapurikar et al., 2006] it was suggested a parallel search over the two sets of Bloom filters/hash tables and the DLA (associated with

the different prefix lengths) for a given input address, which is mentioned to be appropriate for hardware implementations. In this strategy, a final pass is performed to verify if a match occur in any of these data structures and to select the next hop. The same approach could be used for a software-based parallelization by dispatching a thread to search each data structure. However, IPv4 prefix databases have the well-known characteristic that prefixes are not uniformly distributed in the range of valid prefix lengths and, as a consequence, it is more likely that a match occurs to prefixes within lengths that concentrate most of the addresses, i.e., the set of Bloom filter and hash table that stores 24-bit prefixes. Therefore, computing all Bloom filters in parallel may not be efficient because, most of the times, the results from the data structures associated with prefix lengths smaller or greater than 24 bits will not be used. Instead, it is more compute efficient to sequentially query the Bloom filters and the DLA. The other option for TLP, which is used in our approach, is to perform the parallel lookup computation for multiple addresses by assigning one or multiple addresses to each computing thread available. In this way, we can carry out the processing of each address using the most compute efficient algorithm, while we are still able to improve the system throughput by computing the lookup for multiple addresses concurrently. This is possible because the processing of addresses are independent and, as such, there is no synchronization across the computation performed for different addresses. The implementation of the parallelization at this level employed the Open Multi-Processing API (OpenMP) [OpenMP, 2016], which was used to annotate the main algorithm loop that iterates over the input addresses to find their next hops. The specific OpenMP settings used, which led to the better results, were the *dynamic* scheduler and *chunk size* of one.

4.4. Instruction-Level Parallelism (ILP)

The use of ILP is important to take full advantage of the Intel Phi, which is equipped with a 512-bit vector processing unit (see Section 2.1). We used its SIMD instructions to efficiently compute the hash values for multiple input addresses at the same time. The ILP optimization focused on the hashing calculations because it is the most compute intensive stage of the algorithm. The original work [Dharmapurikar et al., 2003] and previous implementations of algorithms employing Bloom filters to the LPM problem [Lim et al., 2014, Ni et al., 2015] do not discuss their decisions and reasons on the hash functions used. Thus, we have decided to implement, vectorize, and evaluate three hash functions: MurmurHash3 [Appleby, 2011] (Murmur), Knuth's multiplicative method [Knuth, 1998] (Knuth), and a hash function named to here as H2 [Mueller, 2006]. Murmur is widely used in the context of Bloom filters, but its original version takes as input a variable-length string. In order to improve its efficiency, we have derived versions of it specialized to work on 32-bit (for IPv4) and 64-bit (for IPv6) integer keys. Knuth is a simple hash function of the form: $h(x) = x \times c \bmod 2^l$, where c should be a multiplier in the order of the hash size 2^l that has no common factors with it. H2 takes as input a key and mixes its bits using a series of bitwise operations, as shown in Algorithm 1. Although simple, the H2 hash function has been shown to be effective in practice [Mueller, 2006].

The hash implementations employed the low-level Intel® Intrinsics API [Intel, 2015] to perform a manual vectorization of all the hash functions. We have also evaluated the use of automatic vectorization available with the Intel® C Compiler, but the manually generated code has proved to be more efficient.

Algorithm 1: Definition of the H2 hash function.

Input : x {A 32-bit unsigned integer key}.
Output: The computed hash value.

1 $x := ((x \gg 16 \oplus x) \times$ 0x45d9f3b
2 $x := ((x \gg 16 \oplus x) \times$ 0x45d9f3b
3 $x := ((x \gg 16 \oplus x)$
4 **return** x

5. Performance Evaluation

This section evaluates the performance of our optimized Bloom filters algorithm both for IPv4 and IPv6. Our evaluations include the analysis of parameter impacts to the performance of the algorithm — namely, the hash functions, FPR and CPE.

5.1. Experimental Setup and Databases

The experiments were performed in a machine equipped with a dual socket Intel® Xeon E5-2640v3 CPU (16 CPU cores with Hyper-Threading), 64 GB of main memory, an Intel® Xeon Phi™ 7120P coprocessor (described in Section 2.1), and CentOS 7 operating system. The source codes were developed using C11 and compiled with the Intel® C Compiler 16.0.3 for both the CPU and the Intel Phi using the -O3 optimization flag.

Database	Location	Originally			After CPE		
		≤ 20	$21-24$	$25-32$	$=20$	$=24$	$=32$
AS65000	-	104,283	516,699	1625	1,048,576	971,555	113,397
DE-CIX	Frankfurt	102,984	535,074	9287	1,048,576	1,007,513	209,488
LINX	London	100,331	519,503	354	1,048,576	982,940	19,863
MSK-IX	Moscow	102,555	528,728	9529	1,048,576	1,004,073	203,360
NYIIX	New York	102,085	528,455	3637	1,048,576	1,000,128	151,391
PTTMetro-SP	Sao Paulo	103,733	544,703	4095	1,048,576	1,024,899	147,201

Table 1. Characteristics of the IPv4 prefix datasets used.

We used 6 real prefix databases for IPv4, whose characteristics are summarized in Table 1. The first database, AS65000, was obtained from [BGP Potaroo, 2016], whereas the remaining databases were downloaded from [RIPE NCC, 2016]. Table 1 presents the amount of addresses in each database and the total number of prefixes before and after performing the CPE to group them into sets of 24-bit and 32-bit long prefixes.

For IPv6, we use the AS65000-V6 database collected from [BGP Potaroo, 2016]. IPv6 is still not widely used and this database has only 31,645 prefixes, which are distributed in 34 distinct prefix lengths. Table 2 shows the effects of applying our DPCPE on the AS65000-V6 database. Note that, for this database, we can not use less than 3 expansion levels, since the amount of memory required becomes prohibitively large. Also, although the algorithm ignores the prefix capture problem (Section 4.2) when computing the levels, its estimates are very close to the actual results.

5.2. The Effect of the Hash Function and False Positive Ratio

The false positive ratio (FPR) is a key aspect for the effectiveness of a Bloom filter because it affects the memory requirements and the number of hash calculations per lookup.

Resulting database	Desired number of distinct lengths	Expansion levels suggested by the algorithm	Estimated number of prefixes	Actual number of prefixes
CPE8	8	$\{24, 29, 33, 38, 40, 44, 48, 64\}$	61,208	60,261
CPE7	7	$\{29, 33, 38, 40, 44, 48, 64\}$	77,700	76,638
CPE6	6	$\{29, 33, 38, 44, 48, 64\}$	104,625	103,094
CPE5	5	$\{33, 38, 44, 48, 64\}$	385,200	380,217
CPE4	4	$\{33, 44, 48, 64\}$	1,185,300	1,166,135
CPE3	3	$\{44, 48, 64\}$	650,417,961	—

Table 2. Results of performing CPE in the AS65000-V6 database. There are a total of 34 distinct lengths and 31,645 unique IPv6 prefixes in AS65000-V6.

We highlight that the FPR does not affect the results of the algorithm, but only the number of times that a value is informed to be in the associated hash table by a Bloom filter without being. When this occurs, the algorithm will unsuccessfully search in the hash table. Probing a hash table consists in traversing a linked list, which may become expensive as the FPR increases. On the other hand, a very low FPR requires a larger number of hash calculations and a high memory utilization. The FPR is determined by three main parameters: the number n of entries stored in the filter, the size m of the filter, and the number k of hash functions used to store/query the filters. Given a n value, the values m and k can be derived as detailed in [Bloom, 1970] to attain a desired FPR.

The trade-off between increasing the hash calculations and the application memory footprint in order to avoid the extra cost of a false positive is complex. Therefore, we have evaluated it experimentally by measuring the performance in million lookups per second (Mlps) of various FPRs and hash function configurations. Hash functions are used in the Bloom filters algorithms for querying the Bloom filters and to address the hash tables associated to each filter. As such, we are able to use combinations of hash functions to compute the multiple hashes within a Bloom filter or the single hash that address a particular hash table. The hash functions used were presented in Section 4, and we employ the AS65000 prefix database and an input address dataset with 2^{26} random addresses.

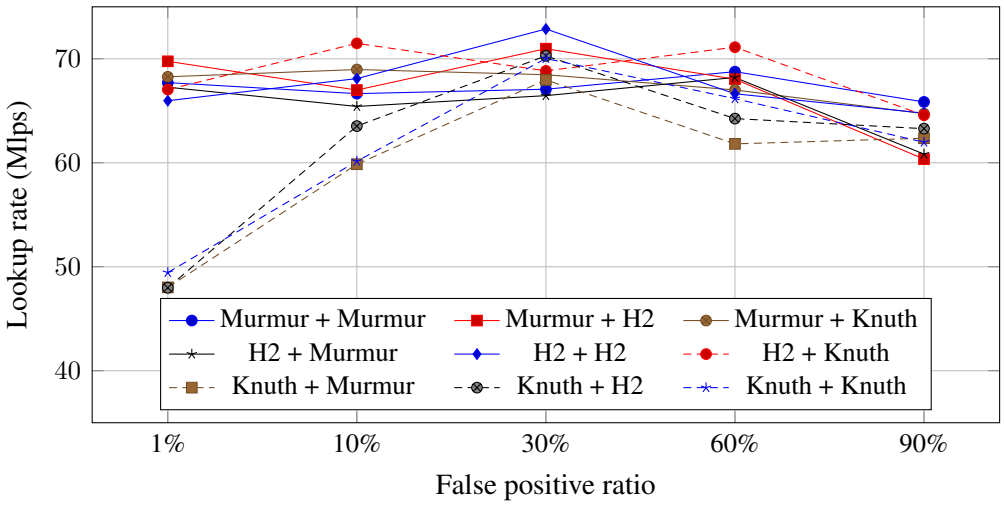


Figure 1. Performance of multiple hash functions and FPRs using 244 threads in the Intel Phi. The "Murmur + H2" entry means Murmur was used within the Bloom filters and H2 was used to address the hash tables.

The results presented in Figure 1 shows that the performance of the application is strongly affected by the FPR and hash functions. As presented, the use of Knuth resulted in a lower average performance as compared to other methods. The reason for the observed results is that this hash function preserves divisibility, e.g., if integer keys are all divisible by 2 or by 4, their hash values will also be. This is a problem in Bloom filters or hash tables in general, where many values will address the same bits in the bit-vector and only a half or a quarter of the buckets will end up being used, respectively. On the other hand, Murmur and H2 are more sophisticated functions that provide better statistical distributions, hence all the configurations using any combination of them attained similar lookup rates. However, the best average performance was reached with 30% of FPR, where the results are less scattered for all hash functions. Furthermore, the best performance was attained when H2 was used in both stages of the algorithm. This occurs, in part, because we are able to reuse the hash calculated to probe the Bloom filter to address the hash table and, as a consequence, hash calculations are saved. Therefore, we use the configuration of 30% of FPR and H2 for both stages in the remaining experiments.

5.3. The Impact of the Input Addresses (Querying) Characteristics to Performance

In order to investigate the effects of the input addresses on the performance, we performed the lookups using pseudo-random generated datasets containing 2^{26} IP addresses with different matching ratios and the AS65000 prefix database. We call *matching ratio* the relation between the number of addresses that matches at least one prefix in the database and the total number of addresses, thus a matching ratio of 80% implies that 20% of the input addresses do not match any prefix in the database and, as such, end up being forwarded to the default route. This evaluation intended to vary the characteristics of the input data and evaluate the algorithms under different configurations.

We ensure that a given address has the same probability to match any prefix stored in the database, and we also filter out all the IETF/IANA reserved IP addresses. Please note that a workload for forwarding could include other characteristics, such as the arrival of packets in bursts. We use a randomized input because it may be considered the worst-case scenario and it is the most commonly method used in previous works. The lookup rates obtained for the *bloomfwd* algorithm on both the CPU and on the Intel Phi are shown in Figure 2.

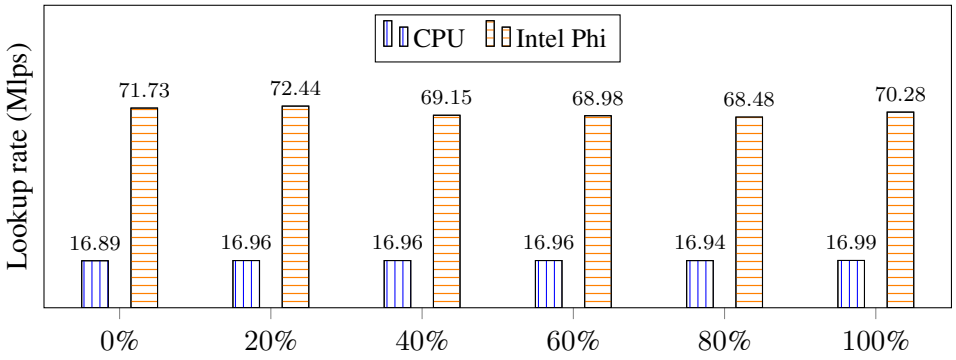


Figure 2. Performance of as matching ratio is varied. The entry 80% means that this percentage of the addresses match with equal probability a prefix in the forwarding table, while 20% of them end up in the default route.

As presented in Figure 2, the input querying dataset has little impact in the overall performance of the application. The reason for that is that the case of an address matching

some prefix in the database *is not necessarily faster* than the case where the address end up in the default route, and vice-versa. For example, consider an address that does not match any prefix in the forwarding table. If no false positives occur, i.e., the two Bloom filters correctly answer not to look in their associated hash tables, the search quickly finishes with one additional memory access to the DLA. However, if for another prefix a false positive occurs in the first, but not in the second Bloom filter, or if there are plenty of values stored in the searched hash table buckets, then this case of matching will likely be *slower* than the former. As such, the speed and good statistical distribution of the hash function to control the FPR and also minimize the number of collisions in the hash tables is an important aspect to limit the variations in the performance as a result of datasets characteristics.

5.4. Scalability and Performance Evaluation: Bloom filters vs. MIHT

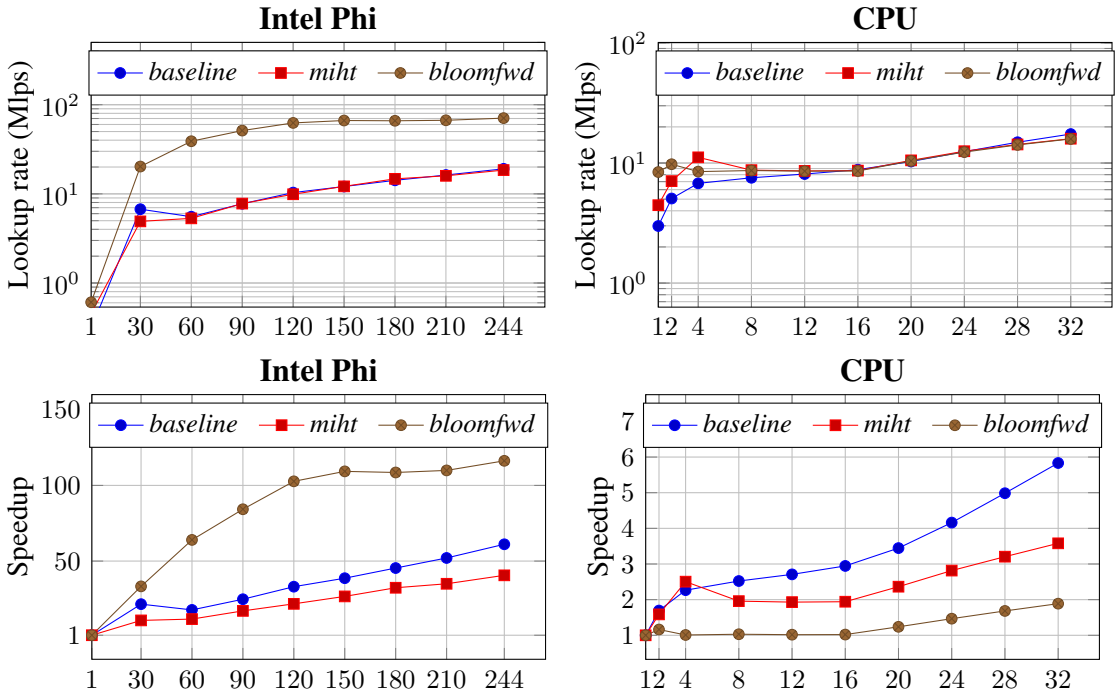


Figure 3. Lookup rate and scalability of the IPv4 algorithms on Intel Phi and CPU using the AS65000 prefix database and the 80% address dataset.

In this section, we evaluate the performance gains of our optimized Bloom filters algorithm for IPv4, which we refer here to as *bloomfwd*, as the number of computing cores used is increased on the Intel Phi and on the CPU. We compare *bloomfwd* to a *baseline* implementation of the Bloom filters algorithm, introduced in Section 4.1, and to the Multi-Index Hybrid Trie (MIHT). The difference between *baseline* and *bloomfwd* is the hash function, i.e., *baseline* uses the standard C *rand* function with no vectorization [Dharmapurikar et al., 2006, Lin et al., 2014]. The MIHT is a state-of-the-art software-based algorithm that has outperformed several other popular IP lookup algorithms [Lim et al., 2014]. Our implementation of MIHT for IPv4 (*miht*) was optimally tuned according to the parameters suggested in the original work for the (16,16)-MIHT. The speedups for the IPv6 dataset are similar and were omitted because of space limitations.

The evaluations used the AS65000 database and an address dataset with 80% of matching ratio. The lookup rates (in log scale) and speedups for both algorithms and processors are presented in Figure 3. As shown, the performance of *miht* ($\approx$ 0.46 Mlps) is better than *baseline* ($\approx$ 0.31 Mlps) for the sequential execution on the Intel Phi. However, as the number of computing threads used increases, the performance gap reduces quickly due to the better scalability of the Bloom filters approach. For instance, the maximum speedup of *baseline* as compared to its sequential counterpart is about 61$\times$, whereas *miht* attains a speedup of up to 40$\times$ when compared to its sequential version. The *bloomfwd*, on the other hand, is the fastest algorithm on a single core and is still able to attain better scalability on the Intel Phi (116$\times$). Also, it is at least 3.7$\times$ faster than the other algorithms. Finally, the difference between the lookup rates of *bloomfwd* and *baseline* highlights the importance of the use of vectorization and the hash function choice to performance.

The analysis of the CPU results show that all algorithms attained very similar lookup rates at scale in a multithread setup, though they attained different speedups. We attribute the similar performance of the algorithms on the CPU to the fact that the memory bandwidth of this processor is much smaller than that of Intel Phi, which limits the scalability of the solutions that are memory-intensive. However, note that in the sequential execution, *bloomfwd* was the fastest, followed by *miht* and then *baseline*. This result further reinforces the importance of the hash function choice to performance and the effectiveness of the H2 hash function.

5.5. Performance of the Bloom filters IP Lookup on IPv4 and IPv6

This section evaluates the best version of the Bloom filter algorithms on IPv4 and IPv6 prefix datasets in the Intel Phi. First, we present the performance for the 5 remaining IPv4 prefix databases presented in Table 1 using the querying input dataset with 80% of matching ratio. The results, presented in Figure 4, show that the performance gains of our *bloomfwd* as compared to the *miht* is about 4$\times$ regardless of the dataset used.

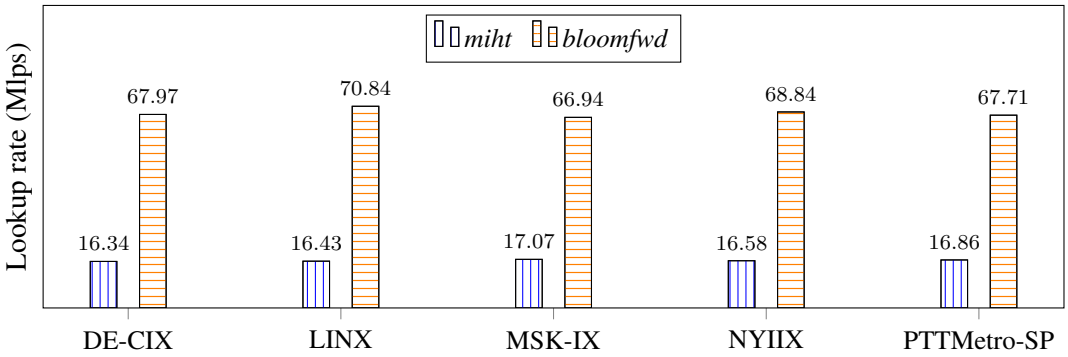


Figure 4. Performance of *bloomfwd* and *miht* for 5 IPv4 prefix datasets.

We further evaluate the performance of the Bloom filters algorithm for IPv6, and the impact of using our DPCPE algorithm. Therefore, we compare the lookup rates of our implementation (*bloomfwd-v6*) with the corresponding version of MIHT for IPv6 (*miht-v6*). The *miht-v6* is equivalent to the (32,32)-MIHT [Lim et al., 2014].

Figure 5 shows the results for the AS65000-V6 with and without the use of DPCPE for multiple expansion levels and 2^{26} random input addresses. As presented, the performance of the *bloomfwd-v6* algorithm is greatly improved by the use of our DPCPE, and the expansion with 7 levels is about 5.9$\times$ faster than the performance without CPE.

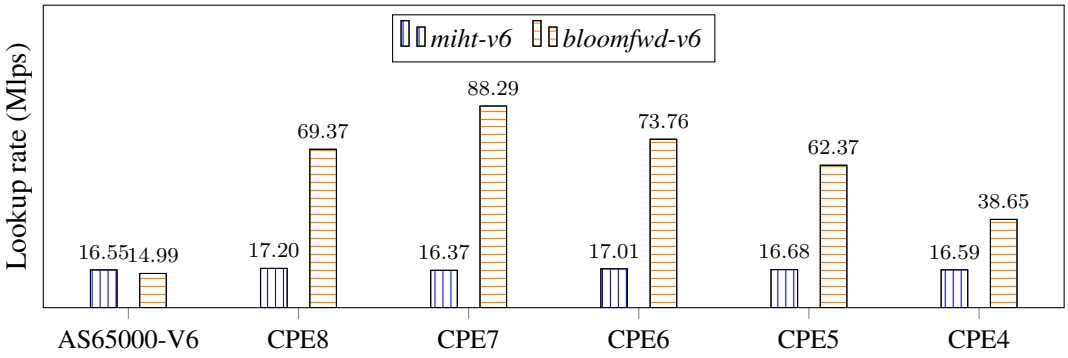


Figure 5. Performance of *bloomfwd* and *miht* for the AS65000-V6 prefix dataset.

This version is also 5.3× faster than the *mith-v6* algorithm. If the expansion value is reduced, however, the performance of the algorithm degrades because of the higher memory demands and search cost. The performance of *miht-v6* is similar in all cases because, in the MIHT, routes are grouped by keys in Priority Tries (PTs), rather than prefix lengths (as in the Bloom filters approach). In other words, the number of distinct prefix lengths in the forwarding table does not directly affect the performance of MIHT.

6. Conclusions and Future Directions

In this work, we have implemented and evaluated the performance of state-of-the-art algorithms for IP lookup (MIHT and Bloom filters approach) in multi-/many-core systems, which is a core operation for efficient packet forwarding in routers. The MIHT is known to be a very efficient sequential algorithm [Lin et al., 2014]. However, it is also very irregular, which typically leads to reduced opportunities for optimized execution on parallel systems. The baseline Bloom filters algorithm, on the other hand, is a more compute intensive and regular algorithm with a less efficient sequential version. Nevertheless, it offers more opportunities for optimizations, for instance, due to SIMD instructions, and it is more scalable with respect to the number of computing cores used. As presented in the results section, our optimized Bloom filters algorithm was able to compute the next hop for 2^{26} IPv4 and IPv6 input addresses, respectively, in a rate of 70.84 Mlps and 88.29 Mlps. Also, the speedup of 116.2× obtained on Intel Phi motivates the use of the proposed techniques, along with many-core technologies, in the construction of efficient software routers. The CPU versions of the algorithms attained good performance but its low memory bandwidth limits the scalability of the algorithms. The good scalability of the Bloom filters approach shows that it may be a better option for devices with a large number of computing cores.

As a future work, we intend to improve the performance of our Bloom filters algorithm by using the CPU and the Intel Phi cooperatively to perform the lookups. We also plan to integrate our optimized algorithm in a complete software router, such as Click or Open vSwitch (OvS).

References

Appleby, A. (2011). MurmurHash3 Hash Function. `https://code.google.com/p/smhasher/wiki/MurmurHash3`.

Asai, H. and Ohara, Y. (2015). Poptrie: A Compressed Trie with Population Count for Fast and Scalable Software IP Routing Table Lookup. In *ACM SIGCOMM Computer Communication Review*, volume 45, pages 57–70. ACM.

BGP Potaroo (2016). BGP Potaroo. `http://bgp.potaroo.net/`.

Bloom, B. H. (1970). Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13(7):422–426.

Dharmapurikar, S., Krishnamurthy, P., and Taylor, D. E. (2003). Longest prefix matching using bloom filters. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 201–212. ACM.

Dharmapurikar, S., Krishnamurthy, P., and Taylor, D. E. (2006). Longest Prefix Matching Using Bloom Filters. *IEEE/ACM Transactions on Networking*, 14(2):397–409.

Fan, L., Cao, P., Almeida, J., and Broder, A. Z. (2000). Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. *IEEE/ACM Transactions on Networking (TON)*, 8(3):281–293.

Intel (2015). Intel Intrinsics Guide. urlhttps://software.intel.com/sites/landingpage/IntrinsicsGuide/.

Jeffers, J. and Reinders, J. (2013). *Intel Xeon Phi coprocessor high-performance programming*. Elsevier Waltham (Mass.), Amsterdam, Boston (Mass.).

Kirsch, A. and Mitzenmacher, M. (2008). Less Hashing, Same Performance: Building a Better Bloom Filter. *Random Structures & Algorithms*, 33(2):187–218.

Knuth, D. E. (1998). *The Art of Computer Programming: Sorting and Searching*, volume 3. Pearson Education.

Lim, H., Lim, K., Lee, N., and Park, K.-H. (2014). On Adding Bloom Filters to Longest Prefix Matching Algorithms. *IEEE Transactions on Computers*, 63(2):411–423.

Lim, H., Yim, C., and Swartzlander Jr, E. E. (2010). Priority Tries for IP Address Lookup. *IEEE Transactions on Computers*, 59(6):784–794.

Lin, C.-H., Hsu, C.-Y., and Hsieh, S.-Y. (2014). A Multi-Index Hybrid Trie for Lookup and Updates. *IEEE Transactions on Parallel and Distributed Systems*, 25(10):2486–2498.

Mueller, T. (2006). H2 Database Engine. `http://h2database.com`.

Ni, S., Guo, R., Liao, X., and Jin, H. (2015). Parallel Bloom Filter on Xeon Phi Many-Core Processors. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 388–405. Springer.

OpenMP (2016). OpenMP API for Parallel Programming, Version 4.0. `http://openmp.org/wp/`.

Rétvári, G., Tapolcai, J., Kőrösi, A., Majdán, A., and Heszberger, Z. (2013). Compressing IP Forwarding Tables: Towards Entropy Bounds and Beyond. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 111–122. ACM.

RIPE NCC (2016). RIPE Network Coordination Centre. `http://data.ris.ripe.net/`.

Ruiz-Sanchez, M., Biersack, E. W., Dabbous, W., et al. (2001). Survey and Taxonomy of IP Address Lookup Algorithms. *IEEE Network*, 15(2):8–23.

Venkatachary, S. and Varghese, G. (1998). Faster IP Lookups using Controlled Prefix Expansion. *PERFORMANCE EVALUATION REVIEW*, 26:1–10.

Yang, T., Xie, G., Li, Y., Fu, Q., Liu, A. X., Li, Q., and Mathy, L. (2015). Guarantee IP Lookup Performance with FIB Explosion. *ACM SIGCOMM Computer Communication Review*, 44(4):39–50.

ST-Drop: Uma Nova Estratégia de Gerenciamento de Buffer em Redes Oportunistas D2D

Michael D. Silva[1], Ivan O. Nunes[1], Raquel A. F. Mini[2], Antonio A. F. Loureiro[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Belo Horizonte – MG – Brasil

[2]Departamento de Ciência da Computação – Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte – MG – Brasil

{micdoug, ivanolive, loureiro}@dcc.ufmg.br, raquelmini@pucminas.br

Abstract. *In D2D opportunistic networks, nodes act as relays for transmitting messages to other nodes according to an opportunistic routing algorithm. In this scenario, each node uses a buffer with limited capacity to store these messages temporarily until they are propagated to a neighbor node according to an opportunistic routing protocol. However, when multiple messages are forwarded in the network, the number of incoming messages may exceed the nodes' capacity, causing a buffer overflow. Therefore, message dropping policies are very important to this problem, because when a message is dropped, there is a chance that other copies of this message still exist in the network. This work proposes a new buffer management algorithm for opportunistic routing in D2D networks named ST-Drop (Space-Time-Drop). We evaluate our solution in three different types of opportunistic routing algorithms: epidemic-based, probabilistic, and social-aware. We conduct simulations using two different publicly available data sources and consider different network traffic loads. Compared to other message drop policies, ST-Drop obtained the highest message delivery ratio in all considered scenarios and the lowest overhead when applied to the state-of-art social-aware and probabilistic routing algorithms, namely, Bubble Rap and Prophet.*

Resumo. *Em redes oportunistas D2D, os nós funcionam como intermediadores na transmissão de mensagens. Nesse cenário, cada nó da rede usa um buffer de memória limitada para armazenar mensagens temporariamente até serem encaminhadas para um nó vizinho de acordo com um algoritmo de roteamento oportunístico. Quando várias mensagens são encaminhadas na rede, o número de mensagens recebidas por um nó pode exceder sua capacidade de armazenamento, causando um estouro de buffer. Assim, políticas de descarte de mensagens são muito importantes para este problema, já que quando uma mensagem é descartada por um nó, existe a possibilidade de outras cópias dessa mensagem existirem na rede. Este trabalho propõe uma nova política de descarte de mensagens chamada ST-Drop (Space-Time-Drop). A solução foi avaliada em três diferentes tipos de algoritmos de roteamento: abordagem epidêmica, probabilístico e ciente de contexto social. Foram realizadas simulações com dois traces de mobilidade disponíveis publicamente e consideradas diferentes cargas de tráfego na rede. Os resultados mostram que a ST-Drop, quando comparada com outras políticas de descarte de mensagens, obteve os melhores valores de taxa de entrega em todos os cenários com o menor nível de sobrecarga na rede quando utilizado em algoritmos cientes de contexto social e probabilísticos, respectivamente BubbleRap e Prophet.*

1. Introdução

Nos últimos anos, houve uma grande evolução dos dispositivos móveis. Tais dispositivos nunca estiveram tão presentes no dia a dia das pessoas, sendo utilizados para as mais diversas tarefas, nas quais destacam-se o acesso a aplicativos de redes sociais, e-mail e Web. Como consequência, o nível de tráfego imposto às redes celulares vem aumentando significativamente. Como a expansão da capacidade e da infraestrutura das redes celulares apresenta alto custo e, as vezes, é até inviabilizada por outros fatores, surge a oportunidade de pesquisar soluções alternativas para o problema. Dentre as soluções propostas, a comunicação direta entre os dispositivos, denominada D2D (*Device-to-Device*) mostra-se como uma solução altamente viável, visto que a estrutura de rede pré-existente é reaproveitada para melhorar o fluxo. Estudos recentes [Aijaz et al. 2013, Andreev et al. 2014, Antonopoulos et al. 2014, Asadi et al. 2014, Bastug et al. 2014, Doppler et al. 2009, Nunes et al. 2016b, Pyattaev et al. 2013, Yang et al. 2013] mostram que a comunicação D2D é uma boa solução para disseminação de dados e soluções para diminuir a carga nas estações rádio-base.

Na comunicação D2D, os dispositivos trocam mensagens quando eles estão suficientemente próximos uns dos outros, ou seja, quando um contato ocorre, que é definido pela mobilidade humana. Assim, as conexões são intermitentes e não se pode assumir uma rota fim-a-fim entre dois dispositivos. Esse tipo de cenário foi discutido primeiramente no contexto de redes DTN (*Delay Tolerant Networks* – Redes Tolerantes a Atrasos), no qual o nó armazena a mensagem recebida em um *buffer* até que ocorra uma oportunidade para transmiti-la. Esse paradigma é conhecido como *store-carry-and-forward* [Fall 2003]. No cenário de comunicação das redes celulares, essa estratégia pode ser utilizada para transmitir conteúdo que não tenha restrições de tempo de entrega, utilizando a comunicação dispositivo a dispositivo (D2D) até que o conteúdo chegue ao seu destino. Dessa forma, as estações base podem delegar a transmissão desse conteúdo via D2D, diminuindo o seu nível de carga. Esse tipo específico de comunicação D2D é frequentemente denominado de Comunicação D2D Multi-Hop, ou Comunicação D2D Oportunista [Li et al. 2014, Nunes et al. 2016b].

Na comunicação D2D oportunista não se pode assumir rotas fim-a-fim e, assim, algoritmos de roteamento usados em redes tradicionais não funcionam nesse cenário. Algoritmos de roteamento oportunístico podem ser classificados em duas categorias [Misra et al. 2016]: a primeira categoria é a dos algoritmos *single-copy* (cópia única), no qual somente uma cópia de cada mensagem é encaminhada na rede e, o maior desafio é decidir quais rotas devem ser exploradas. Alguns exemplos de algoritmos *single-copy* são Direct Delivery [Spyropoulos et al. 2004], First Contact [Jain et al. 2004] e SimBet [Daly and Haahr 2007]. A segunda categoria é a dos algoritmos *multi-copy* (múltiplas cópias), que encaminham simultaneamente na rede múltiplas cópias da mensagem, explorando diferentes rotas D2D. Os algoritmos de roteamento *multi-copy* podem ainda ser classificados como limitados, quando é definido um limite superior de cópias de uma mesma mensagem, e ilimitados caso contrário. Os principais exemplos de algoritmos de roteamento *multi-copy* limitados são *Spray and Wait* [Spyropoulos et al. 2005] e *Spray and Focus* [Spyropoulos et al. 2007]. Alguns exemplos de algoritmos de roteamento *multi-copy* ilimitados são *Epidemic* [Vahdat et al. 2000], *Prophet* [Lindgren et al. 2003] e *Bubble Rap* [Hui et al. 2011].

Em redes D2D oportunistas, os nós precisam cooperar entre si atuando como intermediadores na transmissão de mensagens de outros nós. Para que essa cooperação seja possível, cada nó da rede deve compartilhar seus próprios recursos, alocando memória para armazenar mensagens em um *buffer*, por exemplo. Esse *buffer* é utilizado para armazenar temporariamente as mensagens sendo encaminhadas e sua capacidade é limitada. Por isso, o nível de tráfego na rede, associado às decisões tomadas pelo algoritmo de

roteamento utilizado, podem levar a um estouro de *buffer* do dispositivo. No caso dos algoritmos de roteamento *multi-copy*, políticas de descarte de mensagens representam uma solução viável para esse problema, pois quando uma mensagem é descartada, existe a possibilidade de outras cópias da mensagem existirem na rede [Akestoridis et al. 2014]. No caso de algoritmos de roteamento *single-copy*, esse problema geralmente é amenizado através do uso de outros mecanismos de controle de congestionamento.

O problema de gerenciamento do *buffer* é ainda mais crítico nos algoritmos de roteamento *multi-copy*, pois além de permitirem a existência de múltiplas cópias de uma mensagem, normalmente não existe um dispositivo central com conhecimento global da rede. Sendo assim, os nós não sabem quando uma mensagem é entregue ao destino. Nos algoritmos *multi-copy*, mesmo quando uma mensagem é entregue no destino, suas cópias permanecem na rede até que seu TTL seja excedido ou outro mecanismo da rede as descarte [Bindra and Sangal 2012]. Um mecanismo eficiente de gerenciamento de *buffer* é um requisito fundamental para esse cenário.

Neste trabalho é proposta uma política de gerenciamento de *buffer* para algoritmos de roteamento *multi-copy* em redes oportunistas D2D, denominada *Space-Time Drop* (ST-Drop). Essa política utiliza informações locais para mensurar a cobertura de tempo e espaço de uma mensagem na rede. Resumidamente, a cobertura de espaço de uma mensagem indica o número de dispositivos que carregam essa mensagem e a cobertura de tempo indica o período de tempo que uma mensagem é mantida por um dispositivo, como discutido na Seção 3. A ideia básica é que uma mensagem com maior cobertura de tempo e espaço tem maior probabilidade de já ter sido entregue ao destino, desta forma, ela pode ser descartada primeiro. Experimentos realizados com um *trace* de mobilidade real, contendo 115 dispositivos, e um *trace* sintético, contendo 500 dispositivos, mostram que a política ST-Drop, associada a algoritmos de roteamento oportunista, permite alcançar uma boa taxa de entrega de mensagens com baixo nível de sobrecarga na rede.

Este trabalho está organizado da seguinte forma: a seção 2 discute trabalhos de pesquisa relacionados. A seção 3 apresenta a política de descarte de mensagens ST-Drop, com uma descrição detalhada de seu algoritmo. A seção 4 descreve a metodologia experimental utilizada para validar a ST-Drop, incluindo a descrição dos algoritmos de roteamento oportunistas utilizados, os *traces* de mobilidade e todos os parâmetros de simulação. A seção 5 apresenta os resultados de simulação e os discute. Por fim, a seção 6 apresenta uma discussão final onde são apontadas direções futuras do trabalho.

2. Trabalhos Relacionados

Na comunicação D2D, os dispositivos devem alocar uma porção de memória para ser utilizada como *buffer* de armazenamento temporário das mensagens recebidas por eles. A ideia básica, para evitar situações de sobrecarga no *buffer*, mantendo a eficiência do algoritmo de roteamento oportunista utilizado, consiste na utilização de mecanismos de controle de congestionamento. Em [Silva et al. 2015], é feita uma classificação dos mecanismos de controle de congestionamento, fazendo um estudo extenso sobre políticas de descarte de mensagens utilizadas em redes oportunistas. Nesta seção, são discutidas as principais políticas de descarte de mensagens propostas na literatura.

Em [Lindgren and Phanse 2006], é proposta a política de descarte *Evict Most Forwarded* (MOFO), que se baseia na ideia de que uma mensagem que foi encaminhada um maior número de vezes tem maior chance de já ter sido entregue, mesmo se for descartada localmente. Também é proposta a política *Evict Shortest Life Time First* (SHLI), que se baseia na ideia de que faz mais sentido alocar recursos para mensagens que tenham um maior valor de TTL, pois, intuitivamente, essas mensagens têm maior probabilidade de serem entregues. Já a política *Drop Largest* [Rashid and Ayub 2010] seleciona primeiro

as mensagens com maior tamanho de serem descartadas, pois, ao fazer isso, mais espaço é liberado no *buffer* com menor número de descartes. Outra política básica é a FIFO (*First In First Out*), no qual, como o nome sugere, mensagens que foram recebidas a mais tempo são descartadas primeiro.

Em [Rashid et al. 2013] é proposta a política *Message Drop Control Source Relay* (MDC-SR), no qual o nó deixa de receber mensagens não destinadas a ele quando seu *buffer* alcança um nível de ocupação definido por um parâmetro. Com esta abordagem, a política diminui o descarte de mensagens efetuadas na rede. Experimentos realizados mostram que a política MDC-SR otimiza o desempenho dos algoritmos de roteamento *Epidemic*, *Prophet* e *First Contact* em relação à taxa de entrega e sobrecarga na rede.

Em [Krifa et al. 2008] é proposta a política *Global Knowledge Based Drop* (GBD-Drop), no qual primeiramente é assumido um conhecimento global da rede para decidir quais mensagens devem ser descartadas, levando a um desempenho ótimo. Uma versão distribuída do algoritmo é proposta, no qual são utilizadas técnicas de aprendizagem estatística para aproximar o conhecimento global da rede. Experimentos realizados mostram que tanto a versão centralizada quanto a distribuída alcançam melhores resultados quando comparadas com políticas básicas. A política GDB-Drop pode ainda ser parametrizada para alcançar melhor tempo médio de entrega ou taxa de entrega.

Em [Rashid et al. 2011] é proposta a política de descarte *E-Drop*, no qual mensagens com tamanho maior ou igual à mensagem sendo recebida pelo dispositivo são descartadas primeiro, quando é necessário liberar espaço no *buffer*. No caso em que a mensagem sendo recebida tem tamanho maior que todas as mensagens no *buffer* do dispositivo, o descarte é avaliado seguindo a lógica da política FIFO. Essa política minimiza o número de descarte de mensagens, pois ela vai normalmente remover somente uma mensagem do *buffer* para receber uma nova mensagem. Simulações realizadas com modelos de mobilidade mostram que a política E-Drop tem melhor desempenho que a política MOFO.

Em [Li et al. 2009] é proposta a política *N-Drop*, no qual as mensagens que foram transmitidas um número de vezes maior que uma constante pré-determinada são descartadas primeiro. Simulações realizadas mostram que a política tem desempenho melhor que a política *FIFO* quando utilizada em conjunto com o algoritmo de roteamento *Epidemic*. Já em [Ayub and Rashid 2010] é proposta a política de descarte *T-Drop*, no qual mensagens com tamanho dentro de um intervalo T predefinido são descartadas primeiro. Simulações mostram que a *T-Drop* tem desempenho melhor que a política FIFO quando utilizada em conjunto com os algoritmos de roteamento *Epidemic* e *Prophet*.

Em [Naves et al. 2012] são propostas duas novas políticas de gerenciamento de *buffer*. A primeira é a *Least Recently Forwarded* (LRF), no qual mensagens que foram encaminhadas mais recentemente são descartadas primeiro. A segunda é a *Less Probable Sprayed* (LPS), na qual as mensagens que têm menor probabilidade de ser entregues são descartadas primeiro. Experimentos realizados com *traces* de mobilidade reais, aplicando os algoritmos de roteamento *Epidemic* e *Prophet*, mostram que os algoritmos têm desempenho superior às políticas FIFO e MOFO em relação a taxa de entrega e sobrecarga na rede.

As políticas discutidas nesta seção, em sua grande maioria, se baseiam somente em uma única métrica local, como por exemplo, tempo no qual a mensagem é carregada, número de transmissões, TTL e tamanho da mensagem. A utilização dessas métricas consiste em uma boa estratégia, pois elas são independentes do algoritmo de roteamento utilizado. Porém, a sua utilização de maneira isolada pode não ser suficiente para decidir de forma eficiente quando descartar uma mensagem ou não. A política ST-Drop, proposta

neste trabalho, combina algumas dessas métricas para criar um novo mecanismo de decisão de descarte. Na próxima seção o funcionamento da ST-Drop é descrito com mais detalhes.

3. O Algoritmo ST-Drop

A formulação da política ST-Drop parte do princípio que uma mensagem com uma maior cobertura de tempo e espaço na rede tem maior probabilidade de já ter sido entregue ao destino, sendo assim, ela pode ser descartada primeiro. Baseado nessa ideia, foi definido um coeficiente de espaço S_c que mede a cobertura de espaço de uma mensagem na rede, e o coeficiente de tempo T_c que mede a cobertura de tempo da mensagem na rede. A partir disso, é possível calcular a cobertura de tempo-espaço ST_c usando a seguinte função:

$$ST_c = S_c \cdot T_c. \tag{1}$$

Essa primeira formulação não define de que forma os coeficientes S_c e T_c são calculados. Algoritmos de gerenciamento de *buffer* em redes oportunistas D2D devem preferencialmente ser implementados de forma distribuída, ou seja, devem se restringir a usar somente informações locais. Além disso, a política ST-Drop foi projetada para funcionar de forma independente do algoritmo de roteamento utilizado. Assim, foi definido que a política ST-Drop utilizaria a combinação de métricas locais, utilizadas por outros algoritmos discutidos anteriormente, para calcular os coeficientes S_c e T_c propostos.

O primeiro problema consiste em calcular a cobertura de espaço. Em redes celulares, os contatos são gerados a partir da mobilidade humana. Intuitivamente, pode-se dizer que uma mensagem carregada por uma única pessoa, terá menos oportunidades de ser transmitida, quando comparada com uma mensagem carregada por duas ou mais pessoas, pois ela estará restrita à mobilidade de uma única pessoa. Portanto, pode-se dizer que o número de dispositivos que carregam uma mensagem é um indicador sensato da cobertura de espaço de uma mensagem, pois a cobertura de espaço de uma mensagem é um resultado direto da combinação da mobilidade dos dispositivos que a carregam. Desta forma, a política ST-Drop utiliza a mesma métrica utilizada pela política MOFO para calcular a cobertura de espaço, ou seja, o número de vezes que uma mensagem foi transmitida por um nó. O número de transmissões de uma mensagem é um indicador local do número de dispositivos que carregam uma mensagem.

O segundo problema consiste em calcular a cobertura de tempo. A política FIFO explora o tempo em que as mensagens são carregadas por um nó para decidir qual mensagem descartar primeiro. Ao usar somente essa métrica, uma mensagem criada recentemente pode ser descartada ao invés de uma mensagem com TTL quase expirando. Já a política SHLI utiliza o TTL das mensagens para decidir qual deve ser descartada primeiro. Ao usar somente essa métrica, uma mensagem com valor de TTL relativamente grande pode ser mantida indefinidamente no *buffer* de um nó que tem pouca possibilidade de entregá-la ao destino. Outro problema relacionado com essa abordagem é que em cenários reais as mensagens criadas podem ter valores de TTL iniciais diferentes, o que pode levar a uma mensagem com TTL inicial relativamente baixo a ser penalizada. Com o objetivo de amenizar esses problemas, a política ST-Drop combina as ideias das políticas FIFO e SHLI, normalizando o tempo que a mensagem está sendo carregada CT (*Carrying Time*) pelo seu TTL. Assim, o valor de T_c é definido pela função:

$$T_c = \frac{CT}{TTL}. \tag{2}$$

Com esta formulação é possível calcular a cobertura de tempo e espaço de uma mensagem usando somente informações locais. O Algoritmo 1 apresenta a política ST-Drop, que só é acionada quando não há espaço suficiente no *buffer* para receber uma

nova mensagem. O algoritmo ordena as mensagens em ordem decrescente pelos seus valores de ST_c, criando assim uma fila de descarte. As mensagens com ST_c igual a 0 são retiradas desta fila, o que implica em estas mensagens nunca serem descartadas. O algoritmo considera que descartar uma mensagem com cobertura de tempo e espaço igual a 0 não é justo, pois a mensagem não foi retransmitida para nenhum outro nó neste caso. Após essas etapas, a primeira mensagem da fila é descartada até que seja liberado espaço suficiente para receber a nova mensagem, ou a fila se torne vazia. Se após os descartes foi possível liberar espaço suficiente, o algoritmo retorna uma confirmação para receber a nova mensagem. Caso contrário, o dispositivo deve rejeitar a mensagem que acabou de ser recebida.

Algorithm 1 Algoritmo ST-Drop

Input: msgRecebida
Input: buffer
Output: receber
1: **if** msgRecebida.tamanho $>$ buffer.capacidade **then**
2: **return** false
3: **end if**
Ordena as mensagens em ordem decrescente por ST_c :
4: fila_descarte $\leftarrow$ ordenar(buffer)
Remove as mensagens com $ST_c = 0$
5: fila_descarte.filtrar()
Descarta mensagens até que seja liberado espaço suficiente
6: tamanhoRecebido $\leftarrow$ msgRecebida.tamanho
7: disponivel $\leftarrow$ buffer.espacoDisponivel
8: **while** disponivel $<$ tamanhoRecebido e não fila_descarte.vazia **do**
9: msg $\leftarrow$ fila_descarte.removerPrimeira()
10: buffer.descartar(msg)
11: disponivel $\leftarrow$ ++msg.tamanho
12: **end while**
13: **if** disponivel $<$ tamanhoRecebido **then**
14: **return** false
15: **else**
16: **return** true
17: **end if**

4. Metodologia de Simulação

Esta seção descreve as simulações realizadas para avaliar a política ST-Drop. Primeiramente, são descritas as métricas de avaliação de comunicação oportunista D2D que foram utilizadas. Depois são apresentados os algoritmos de roteamento aplicados nas simulações e são descritos os dois *traces* de mobilidade utilizados para simular a movimentação dos nós e, consequentemente, a geração de contatos. Por fim, são expostos e discutidos os parâmetros e configurações utilizados.

4.1. Métricas

Para comparar o desempenho da política de descarte ST-Drop com as outras políticas, foram utilizadas as seguintes métricas:

- **Taxa de Entrega**: avalia o percentual de mensagens entregues ao destino ao longo do tempo;

- **Número de Transmissões**: avalia o número de retransmissões por mensagem criada na rede, ou seja, o número de transmissões D2D que cada algoritmo executou ao longo do tempo, normalizado pelo número de mensagens criadas.

Políticas de descarte de mensagens para redes D2D devem idealmente alcançar um bom custo benefício considerando essas métricas, ou seja, devem alcançar um alto nível de taxa de entrega com baixo número de transmissões. As mensagens entregues ao destino com sucesso representam aquelas mensagens que as estações base não precisarão transmitir, economizando, assim, recursos computacionais do dispositivo (e.g., processamento e energia) e do canal de comunicação. Um grande número de transmissões de mensagens pode impactar negativamente a experiência dos usuários, por exemplo, aumentando o gasto de energia dos dispositivos.

4.2. Algoritmos de Roteamento Oportunista

Para avaliar o desempenho da política ST-Drop, foram utilizados algoritmos de roteamento que se baseiam em três abordagens distintas, permitindo, assim, avaliar o comportamento da solução proposta quando integrada a diferentes algoritmos de roteamento. O primeiro algoritmo avaliado foi o *Epidemic* [Vahdat et al. 2000], também conhecido como *Flooding*. Este algoritmo é bastante simples, visto que não se baseia em funções de utilidade ou métricas similares. Nesse algoritmo os nós da rede a cada encontro trocam todas mensagens não comuns entre eles. Quando se considera um cenário onde os dispositivos da rede têm *buffers* com capacidade ilimitada, o algoritmo *Epidemic* alcança o valor máximo possível de taxa de entrega, porém também alcança um número muito alto de transmissões. Porém, quando se considera um cenário mais realista, no qual os dispositivos têm *buffers* com capacidade de armazenamento limitada, esse algoritmo por si só pode não alcançar o melhor percentual de taxa de entrega possível.

O segundo algoritmo de roteamento considerado nas simulações é o *Prophet* [Lindgren et al. 2003]. Esse algoritmo utiliza o histórico de contatos dos nós para calcular a probabilidade de um determinado elemento entregar uma mensagem. Quando dois nós entram em contato, eles trocam somente as mensagens as quais o outro nó tem maior probabilidade de entrega. A ideia básica do *Prophet* consiste em considerar que um par de nós que se encontrou recentemente tem grande probabilidade de se encontrar novamente.

O terceiro algoritmo de roteamento utilizado é o Bubble Rap [Hui et al. 2011]. Este algoritmo se baseia nos conceitos sociais de comunidade e popularidade. Nesse algoritmo, cada nó é associado a pelo menos uma comunidade social, que, por sua vez, é definida como um conjunto de nós densamente conectados (um grupo de nós que tem mais contatos entre si do que com outros). A popularidade de um nó é calculada a partir do número de contatos distintos que um determinado nó teve ao longo do tempo. A ideia básica do Bubble Rap é encaminhar as mensagens para nós que tenham um maior nível de popularidade global até que a mensagem seja alcançada por um membro da comunidade destino (uma das comunidades associadas ao nó destino). A partir deste momento, a mensagem é então encaminhada somente entre membros da comunidade destino com base no nível de popularidade local (considerando somente os membros da comunidade), até que a mensagem chegue ao destino. As comunidades e valores de centralidade, necessários para utilizar o Bubble Rap, foram obtidos a partir dos algoritmos *Distributed K-Clique Community Detection* [Hui et al. 2007] e *C-Window* [Hui et al. 2011], da forma que foram definidos na formalização original do algoritmo de roteamento.

Como foi discutido, foram utilizados algoritmos de roteamento bem distintos, baseados em estratégias de epidemia (inundação), funções de probabilidade e ciente de contexto social. Assim, pôde-se avaliar o impacto de diferentes tipos de roteamento no desempenho da política ST-Drop.

Tabela 1. Níveis de Tráfego por Cenário

Cenário		Mensagens/Dia	Total
NCCU	50	58	522
	100	115	1035
SWIM	50	250	1250
	100	500	2500

4.3. *Traces* de Mobilidade

Foram utilizados dois *traces* de mobilidade para simular a movimentação dos nós da rede. O primeiro é o *trace* NCCU [Tsai and Chan 2015], que utiliza dados reais coletados a partir de um experimento realizado com 115 estudantes em um campus universitário durante 15 dias. O segundo é um *trace* sintético gerado a partir do modelo de mobilidade *Small World in Motion* (SWIM) [Kosta et al. 2014]. Esse *trace* se baseia em um modelo de mobilidade tido com estado da arte, que captura a existência de comunidades sociais que influenciam a mobilidade humana. O *trace* utilizado simula a movimentação de 500 pessoas por um período de 11 dias. A utilização dos dois *traces* permite avaliar a ST-Drop sob duas escalas de rede diferentes.

4.4. Execução

Para simular a utilização dos algoritmos de roteamento com diferentes políticas de descarte de mensagens, foi utilizado o simulador de redes oportunistas *The One* [Keränen et al. 2009], um simulador de eventos discreto que não oferece suporte para utilização de algoritmos de gerenciamento de *buffer* personalizados, nem para o algoritmo de roteamento Bubble Rap. Assim, foi necessário implementar esses algoritmos[1] para realizar as simulações.

Para avaliar a política ST-Drop, foi necessário primeiramente definir a capacidade do *buffer* dos nós da rede e um modelo de tráfego de mensagens. Como discutido em [Grasic and Lindgren 2012], não existe um padrão bem definido em relação à definição de modelos de tráfego e mensagens para testar soluções neste cenário. Portanto, foram escolhidos valores que permitissem a correta avaliação dos resultados. A capacidade do *buffer* dos nós foi definida como 1 GB. As mensagens têm sete dias de TTL inicial e são geradas de forma aleatória com distribuição uniforme ao longo do tempo de simulação. O tamanho das mensagens é definido dentro do intervalo [50 MB, 100 MB] por uma distribuição de probabilidade uniforme. A origem e o destino de cada mensagem também são selecionados a partir de uma distribuição de probabilidade uniforme sobre os nós da rede.

Foram considerados três níveis de tráfego de mensagens relativos ao número de dispositivos de cada cenário. O primeiro gera a cada dia um número de mensagens igual a 50% do número de nós, o segundo gera 75% e o último gera 100%. Os resultados do cenário com 75% de nível de tráfego não são expostos, devido à limitação de espaço, porém isso não impacta na análise apresentada. A Tabela 1 apresenta o número de mensagens geradas nos níveis de tráfego de 50% e 100%.

Cada combinação de cenário e nível de tráfego foi simulada 10 vezes com diferentes sementes de geração de origem, destino e tamanho de mensagens. A taxa de entrega e o número de transmissões de cada cenário foram calculados e são apresentados utilizando um intervalo de confiança de 95%

[1] O código fonte das alterações está disponível em `https://github.com/micdoug/the-one/tree/58e207ac44d5012fac5d103da781d30266164216`

5. Resultados

Os resultados de simulação dos cenários NCCU100 e NCCU50 estão representados nas Figuras 1 e 2, respectivamente. Os resultados dos cenários SWIM100 e SWIM50 estão representados nas Figuras 3 e 4, respectivamente. Os valores referentes à taxa de entrega são apresentados em função do tempo de vida das mensagens (até 7 dias nos experimentos realizados) e não do tempo total de simulação. Desta forma é possível ter uma visão mais clara do impacto das políticas na taxa de entrega de cada mensagem. Já os valores referentes ao número de transmissões efetuadas são apresentados considerando todo o tempo de simulação, já que a transmissão de cada mensagem impacta o *overhead* da rede como um todo.

Ao analisar os resultados percebe-se que a evolução da taxa de entrega e do número de transmissões exibe o mesmo comportamento mesmo em diferentes níveis de tráfego. Isso mostra que as políticas mantém um comportamento estável sob diferentes níveis de carga. Pode-se notar também que o trace SWIM exibe um ambiente mais desafiador em relação à entrega de mensagens quando comparado com o trace NCCU. Isso ocorre devido à características referentes ao próprio padrão de contatos exibido pelo trace. Desta forma, os valores de taxa de entrega para todas as políticas no trace SWIM são bem próximos, e, tecnicamente equivalentes quando considera-se o intervalo de confiança.

Os resultados de taxa de entrega no trace NCCU mostram que a política SHLI tem vantagem expressiva nos primeiros três dias do tempo de vida das mensagems, considerando-se os três algoritmos de roteamento utilizados. Isso se deve ao fato desta política eliminar mensagens com valores de TTL altos. Desta forma, a maioria das mensagens entregues se concentram neste espaço de valores de TTL baixos. Isso é evidenciado pelo fato da política SHLI não apresentar um aumento significativo em sua taxa de entrega após o período inicial de três dias. As políticas E-Drop, FIFO e MOFO têm desempenho abaixo das políticas SHLI e ST-Drop para praticamente qualquer valor de TTL considerado, nos três algoritmos de roteamento considerados. A política ST-Drop exibe um crescimento progressivo de taxa de entrega, alcançando o melhor desempenho ao final da simulação para os três algoritmos de roteamento testados. As outras políticas de descarte, porém, alcançam seu valor máximo de taxa de entrega rapidamente, exibindo um comportamento de estabilização rápida.

O crescimento progressivo da taxa de entrega ao utilizar a ST-Drop é consequência do fato da política não descartar mensagens com coeficiente de tempo-espaço igual a 0. Na formulação do algoritmo, definiu-se que o coeficiente de espaço é medido pelo número de vezes que uma mensagem foi retransmitida. Sendo assim, quando uma mensagem ainda não foi retransmitida por um nó, ela não será descartada. Desta forma, garante-se que pelo menos uma cópia da mensagem será mantida na rede, e, as cópias mantidas tendem a se concentrar em nós situados no final do caminho traçado pela mensagem. Tem-se assim uma aproximação do mecanismo de encaminhamento tradicional em redes *"forward-and-drop"*, no qual, ao encaminhar uma mensagem, o nó a exclui de sua memória. Esse efeito torna-se mais forte à medida que o nível de sobrecarga na rede aumenta. Como as mensagens são encaminhadas por caminhos cada vez maiores e não são descartadas completamente da rede, as mensagens continuam sendo entregues mesmo quando considera-se valores de TTL mais altos. As outras políticas de descarte tendem a realizar um descarte precoce das mensagens quando a rede exibe níveis de sobrecarga altos, o que explica a estabilização rápida.

Os resultados de número de transmissões mostram que a política ST-Drop tem o melhor desempenho quando utilizada com os algoritmos de roteamento BubbleRap e Prophet em ambos traces. Em contrapartida, quando utilizada com o algoritmo Epidemic ela exibe o maior número de transmissões entre as políticas avaliadas. Isso também é ex-

plicado pelo efeito *"forward-and-drop"*. O algoritmo Epidemic não define uma restrição em relação aos caminhos que uma mensagem pode percorrer na rede. Como a ST-Drop exerce o efeito de empurrar a mensagem para outros nós, as mensagens tendem a percorrer caminhos cada vez maiores, mesmo quando já foram entregues. Este efeito pode ser amenizado com a utilização de um mecanismo de eliminação de mensagens já entregues, como por exemplo o VACCINE [Haas and Small 2006]. Os algoritmos de roteamento BubbleRap e Prophet criam naturalmente uma fronteira no caminho das mensagens definida pelos seus critérios de encaminhamento, desta forma a política ST-Drop funciona muito bem nestes dois cenários, apresentando valores de números de transmissões bem mais baixos, quando comparada com as outras políticas.

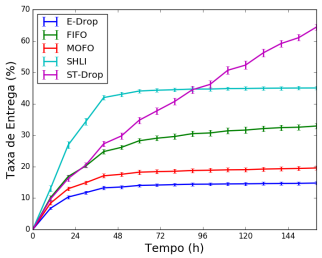
No geral, a política ST-Drop obteve os melhores resultados de taxa de entrega para todos os três algoritmos de roteamento. Em relação ao número de transmissões, a política ST-Drop obteve os melhores resultados em todos os cenários com os algoritmos de roteamento Prophet e Bubble Rap. A política SHLI obteve melhores resultados para valores de tempo de entrega menores, pois nessa política, as mensagens com maior tempo de entrega são descartadas primeiro, levando a um comportamento esperado. Porém quando se considera todo o intervalo de tempo de entrega avaliado, a política ST-Drop obteve melhores resultados que a SHLI.

Vale a pena destacar que, no algoritmo Prophet, a política ST-Drop obteve um número de transmissões muito menor quando comparada às outras políticas. Em relação ao algoritmo de roteamento Bubble Rap, os resultados foram ainda melhores, pois a política ST-Drop alcançou um número de transmissões até três vezes menor que a segunda melhor política. Essa última observação é extremamente importante, pois, atualmente os algoritmos de roteamento oportunista baseados em contexto social são considerados como o estado da arte. A afirmação pode ser confirmada quando comparamos os valores de taxa de entrega e número de transmissões com os três algoritmos de roteamento: Epidemic, Prophet e Bubble Rap. Com ambos *traces* de mobilidade, o Bubble Rap (estratégia baseada no contexto social) obteve os melhores resultados de taxa de entrega com menor número de transmissões, especialmente quando utilizado em conjunto com a ST-Drop.

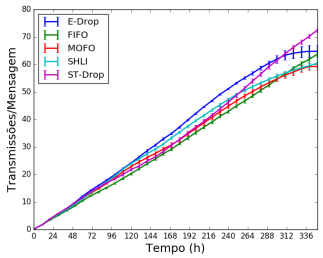
6. Conclusão

Este trabalho propôs uma nova política de gerenciamento de *buffer* em redes oportunistas D2D chamada Space-Time-Drop (ST-Drop). O princípio dessa política é que uma mensagem com maior cobertura de tempo e de espaço na rede tem maior probabilidade de já ter sido entregue, portanto pode ser descartada primeiro. A cobertura de tempo e espaço é calculada a partir da combinação de ideias de políticas mais simples, usando somente informações locais. Assim, a solução pode ser facilmente implantada em ambientes distribuídos. A política ST-Drop foi avaliada a partir de simulação com os algoritmos de roteamento Epidemic, Prophet e Bubble Rap, no qual foram utilizados três níveis diferentes de tráfego. Os resultados mostraram que a política ST-Drop obteve os melhores valores de taxa de entrega com os três algoritmos de roteamento e número de transmissões extremamente mais baixo com os algoritmos de roteamento Prophet e Bubble Rap. Considerando todas as combinações de algoritmos de roteamento oportunista e políticas de gerenciamento de buffer avaliadas, a combinação do Bubble Rap com a política ST-Drop obteve, em todos os experimentos, o melhor custo benefício, ou seja, a melhor razão entre taxa de entrega e número de transmissões. Esse resultado mostra que a política ST-Drop contribui significativamente para melhorar a relação custo-benefício de redes oportunistas D2D.

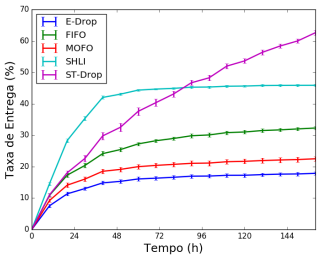
Como trabalho futuro, a política ST-Drop deve ser aplicada a outros algoritmos de roteamento oportunista D2D. Como a política ST-Drop obteve bons resultados com um algoritmo baseado em contexto social, seria interessante aplicá-la a outros algoritmos de



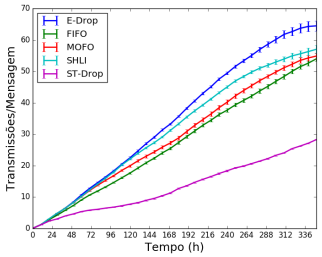
(a) Entrega com Epidemic



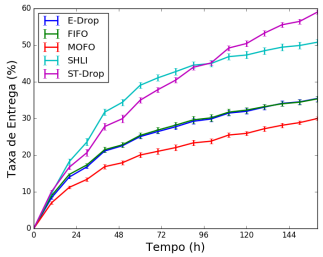
(b) Transmissões com Epidemic



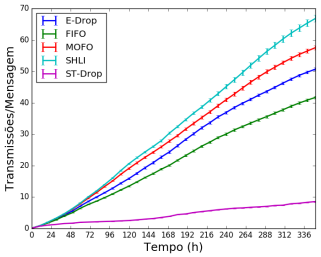
(c) Entrega com Prophet



(d) Transmissões com Prophet

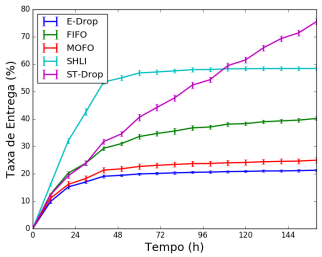


(e) Entrega com BubbleRap

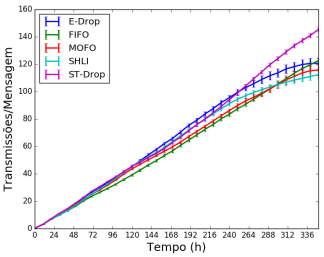


(f) Transmissões com BubbleRap

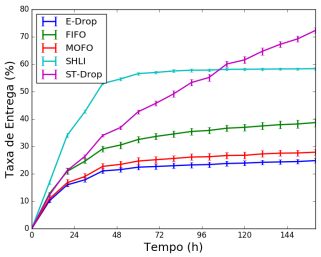
Figura 1. Comparação de políticas de descarte de mensagens no trace NCCU com tráfego de 100% (NCCU100)



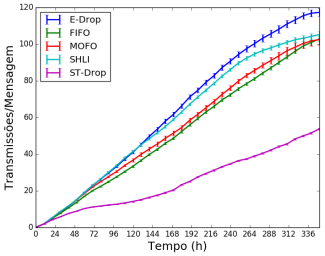
(a) Entrega com Epidemic



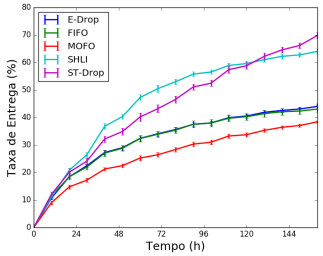
(b) Transmissões com Epidemic



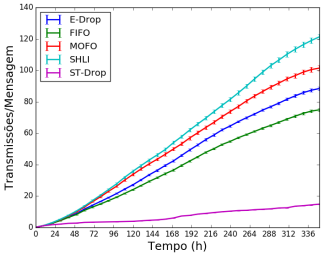
(c) Entrega com Prophet



(d) Transmissões com Prophet



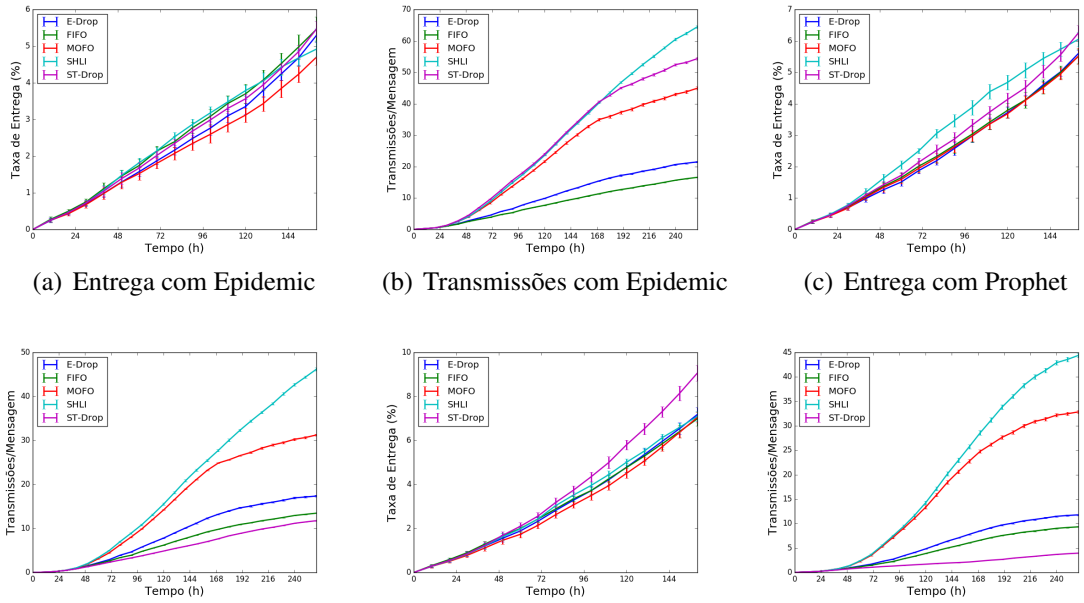
(e) Entrega com BubbleRap



(f) Transmissões com BubbleRap

Figura 2. Comparação de políticas de descarte de mensagens no trace NCCU com tráfego de 50% (NCCU50)

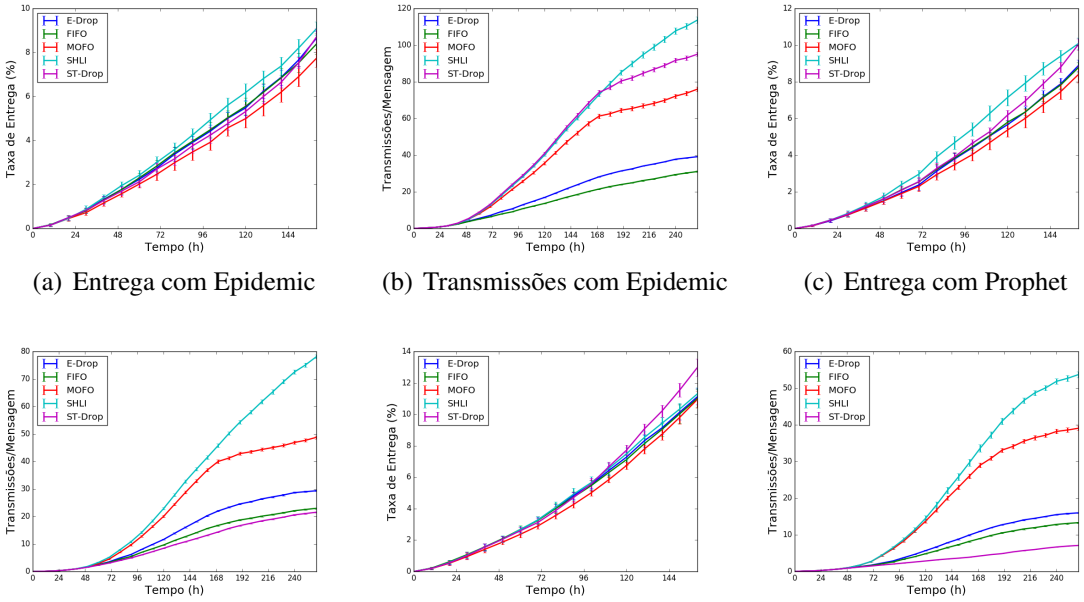
roteamento recentes que utilizam abordagem semelhante como, por exemplo, o GroupsNet [Nunes et al. 2016a, Nunes et al. 2016c]. É importante destacar que também existe a possibilidade de explorar outras métricas para calcular a cobertura de tempo e espaço das mensagens. Informações obtidas a partir da análise de contexto do dispositivo, como nível de bateria e informação espacial (GPS), podem ser utilizadas para melhorar o desempenho



(a) Entrega com Epidemic (b) Transmissões com Epidemic (c) Entrega com Prophet

(d) Transmissões com Prophet (e) Entrega com BubbleRap (f) Transmissões com BubbleRap

Figura 3. Comparação de políticas de descarte de mensagens no trace SWIM com tráfego de 100% (SWIM100)



(a) Entrega com Epidemic (b) Transmissões com Epidemic (c) Entrega com Prophet

(d) Transmissões com Prophet (e) Entrega com BubbleRap (f) Transmissões com BubbleRap

Figura 4. Comparação de políticas de descarte de mensagens no trace SWIM com tráfego de 50% (SWIM50)

da política ST-Drop.

Referências

Aijaz, A., Aghvami, H., and Amani, M. (2013). A survey on mobile data offloading: technical and business perspectives. *IEEE Wireless Communications*, 20(2):104–112.

Akestoridis, D.-G., Papanikos, N., and Papapetrou, E. (2014). Exploiting social preferences for congestion control in opportunistic networks. In *2014 IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 413–418. IEEE.

Andreev, S., Pyattaev, A., Johnsson, K., Galinina, O., and Koucheryavy, Y. (2014). Cellular traffic offloading onto network-assisted device-to-device connections. *IEEE Communications Magazine*, 52(4):20–31.

Antonopoulos, A., Kartsakli, E., and Verikoukis, C. (2014). Game theoretic d2d content dissemination in 4g cellular networks. *IEEE Communications Magazine*, 52(6):125–132.

Asadi, A., Wang, Q., and Mancuso, V. (2014). A survey on device-to-device communication in cellular networks. *IEEE Communications Surveys & Tutorials*, 16(4):1801–1819.

Ayub, Q. and Rashid, S. (2010). T-drop: An optimal buffer management policy to improve qos in dtn routing protocols. *Journal of Computing*, 2(10):46–50.

Bastug, E., Bennis, M., and Debbah, M. (2014). Living on the edge: The role of proactive caching in 5g wireless networks. *IEEE Communications Magazine*, 52(8):82–89.

Bindra, H. S. and Sangal, A. (2012). Need of removing delivered message replica from delay tolerant network-a problem definition. *International Journal of Computer Network and Information Security*, 4(12):59.

Daly, E. M. and Haahr, M. (2007). Social network analysis for routing in disconnected delay-tolerant manets. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 32–40. ACM.

Doppler, K., Rinne, M., Wijting, C., Ribeiro, C. B., and Hugl, K. (2009). Device-to-device communication as an underlay to lte-advanced networks. *IEEE Communications Magazine*, 47(12):42–49.

Fall, K. (2003). A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34. ACM.

Grasic, S. and Lindgren, A. (2012). An analysis of evaluation practices for dtn routing protocols. In *Proceedings of the seventh ACM international workshop on Challenged networks*, pages 57–64. ACM.

Haas, Z. J. and Small, T. (2006). A new networking model for biological applications of ad hoc sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 14(1):27–40.

Hui, P., Crowcroft, J., and Yoneki, E. (2011). Bubble rap: Social-based forwarding in delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 10(11):1576–1589.

Hui, P., Yoneki, E., Chan, S. Y., and Crowcroft, J. (2007). Distributed community detection in delay tolerant networks. In *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, page 7. ACM.

Jain, S., Fall, K., and Patra, R. (2004). *Routing in a delay tolerant network*, volume 34. ACM.

Keränen, A., Ott, J., and Kärkkäinen, T. (2009). The one simulator for dtn protocol evaluation. In *Proceedings of the 2nd international conference on simulation tools and techniques*, page 55. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Kosta, S., Mei, A., and Stefa, J. (2014). Large-scale synthetic social mobile networks with swim. *IEEE Transactions on Mobile Computing*, 13(1):116–129.

Krifa, A., Barakat, C., and Spyropoulos, T. (2008). Optimal buffer management policies for delay tolerant networks. In *2008 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 260–268. IEEE.

Li, Y., Wu, T., Hui, P., Jin, D., and Chen, S. (2014). Social-aware d2d communications: qualitative insights and quantitative analysis. *IEEE Communications Magazine*, 52(6):150–158.

Li, Y., Zhao, L., Liu, Z., and Liu, Q. (2009). N-drop: congestion control strategy under epidemic routing in dtn. In *Proceedings of the 2009 international conference on wireless communications and mobile computing: connecting the world wirelessly*, pages 457–460. ACM.

Lindgren, A., Doria, A., and Schelén, O. (2003). Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE mobile computing and communications review*, 7(3):19–20.

Lindgren, A. and Phanse, K. S. (2006). Evaluation of queueing policies and forwarding strategies for routing in intermittently connected networks. In *2006 1st International Conference on Communication Systems Software & Middleware*, pages 1–10. IEEE.

Misra, S., Saha, B. K., and Pal, S. (2016). Opportunistic mobile networks.

Naves, J. F., Moraes, I. M., and Albuquerque, C. (2012). Lps and lrf: Efficient buffer management policies for delay and disruption tolerant networks. In *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*, pages 368–375. IEEE.

Nunes, I. O., Celes, C., de Melo, P. O., and Loureiro, A. A. (2016a). Groups-net: Group meetings aware routing in multi-hop d2d networks. *arXiv preprint arXiv:1605.07692*.

Nunes, I. O., de Melo, P. O. S. V., and Loureiro, A. A. F. (2016b). Leveraging d2d multihop communication through social group meeting awareness. *IEEE Wireless Communications*, 23(4):12–19.

Nunes, I. O., de Melo, P. O. V., and Loureiro, A. A. F. (2016c). Groups-net: Roteamento ciente de encontros de grupos em redes móveis d2d. In *Proceedings of the Brazilian Symposium on Computer Networks and Distributed Systems*, Salvador, Bahia.

Pyattaev, A., Johnsson, K., Andreev, S., and Koucheryavy, Y. (2013). Proximity-based data offloading via network assisted device-to-device communications. In *Vehicular Technology Conference (VTC Spring), 2013 IEEE 77th*, pages 1–5. IEEE.

Rashid, S. and Ayub, Q. (2010). Efficient buffer management policy dla for dtn routing protocols under congestion. *international Journal of computer and Network Security*, 2(9):118–121.

Rashid, S., Ayub, Q., Zahid, M. S. M., and Abdullah, A. H. (2011). E-drop: An effective drop buffer management policy for dtn routing protocols. *Int. J. Computer Applications*, pages 118–121.

Rashid, S., Ayub, Q., Zahid, M. S. M., and Abdullah, A. H. (2013). Message drop control buffer management policy for dtn routing protocols. *Wireless personal communications*, 72(1):653–669.

Silva, A. P., Burleigh, S., Hirata, C. M., and Obraczka, K. (2015). A survey on congestion control for delay and disruption tolerant networks. *Ad Hoc Networks*, 25:480–494.

Spyropoulos, T., Psounis, K., and Raghavendra, C. S. (2004). Single-copy routing in intermittently connected mobile networks. In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pages 235–244. IEEE.

Spyropoulos, T., Psounis, K., and Raghavendra, C. S. (2005). Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259. ACM.

Spyropoulos, T., Psounis, K., and Raghavendra, C. S. (2007). Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on*, pages 79–85. IEEE.

Tsai, T.-C. and Chan, H.-H. (2015). Nccu trace: social-network-aware mobility trace. *IEEE Communications Magazine*, 53(10):144–149.

Vahdat, A., Becker, D., et al. (2000). Epidemic routing for partially connected ad hoc networks.

Yang, M. J., Lim, S. Y., Park, H. J., and Park, N. H. (2013). Solving the data overload: Device-to-device bearer control architecture for cellular data offloading. *IEEE Vehicular Technology Magazine*, 8(1):31–39.

Trilha Principal do SBRC 2017
Sessão Técnica 7
Redes sem Fio I

Um Modelo de Largura de Banda Flexível para Redes de Rádios Cognitivos Baseadas em Prioridade

Marcos R. M. Falcão[1], Andson M. Balieiro[2], Kelvin L. Dias[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
50670-901– Recife – PE – Brasil

[2] Universidade de Pernambuco (UPE) – Campus Garanhuns

{mrmf,kld}@cin.ufpe.br, andson.balieiro@upe.br

Abstract. *Classic two-priority Cognitive Radio Networks (CRNs) models composed of Primary and Secondary Users (PUs and SUs) are not capable of analyzing SUs with different Quality of Service (QoS) requirements. Recently, few authors have designed three-layered models, but for specific bandwidth (BW) arrangements among the user layers. This paper proposes a continuous time Markov chain model that encompasses all possible BW requirements, which was assessed in terms of blocking and forced termination probabilities. The results show that prioritization highly influences on the overall performance, but the system's BW disposal may also significantly impact on the resource allocation process.*

Resumo. *Modelos de Redes de Rádios Cognitivos (RRCs) compostos por duas prioridades: usuários primários (UPs) e secundários (USs), não possibilitam a análise de USs com diferentes requisitos de Qualidade de Serviço (QoS). Recentemente, modelos com três classes de prioridades têm sido propostos, mas limitados na relação de largura de banda (BW) entre as classes. Este artigo propõe um modelo baseado em Cadeia de Markov de Tempo Contínuo (CMTC) e flexível na BW para RRCs com três níveis de prioridade. As probabilidades de bloqueio e terminação forçada dos USs são avaliadas e os resultados mostram que não somente a priorização influencia no desempenho da RRC, mas também a disposição de BW configurada nas classes.*

1. Introdução

Em Redes de Rádios Cognitivos (RRCs), os usuários secundários (USs) acessam, de forma oportunista, o espectro temporariamente não utilizado pelos usuários primários (UPs) [Liang et al. 2001]. Dado que o ecossistema sem fio compreende aplicações heterogêneas (ex. *streaming* de vídeo, jogos *online*, *web browsing* e *internet banking*) que têm diferentes requisitos de Qualidade de Serviço (QoS), o suporte a múltiplas classes de serviço é uma característica esperada para RRCs. Padrões tais como o IEEE 802.11p para Redes Veiculares Ad Hoc (VANETs) têm sido preparados para atender o montante de largura de banda requerida pelo tráfego de informação/entretenimento, que pode colidir com tipos vitais, como aqueles de aplicações de segurança para motoristas. Assim, para garantir a QoS dos níveis de tráfego mais relevantes, os padrões têm utilizado múltiplas prioridades para gerenciar os diferentes tipos de fluxos na rede. Seguindo esta abordagem,

diversos autores têm adotado classes de prioridade em RRCs, considerando duas [Li et al. 2012; Jiao et al. 2014] ou três prioridades [Chu et al. 2014; Chu et al. 2015].

Entretanto, muitos autores têm proposto modelos inflexíveis quanto à largura de banda requerida pelas classes, i.e., permitem apenas que o UP tenha requisito de largura de banda estritamente maior que o US [Chu et al. 2014; Chu et al. 2015] ou o oposto [Li et al. 2012]. Assim, cenários onde: (1) uma rede inteligente de energia (*smart grid*) é implantada como rede secundária e sinais de TV digital ATSC com 6 MHz de largura de banda compõem a rede primária; e (2) uma rede de streaming multimídia para casas conectadas (ex. usando o padrão IEEE 802.11af) é a rede secundária e sinais de TV analógica NTSC com 100 MHz de largura de banda formam o sistema primário não são possíveis de serem analisados utilizando um único modelo [Li et al. 2012, Jiao et al. 2014; Chu et al. 2014; Chu et al. 2015]. Devido a esta limitação, um sistema mais flexível em termos de largura de banda entre UPs e USs com três camadas de prioridades é proposto neste artigo. Este modelo classifica a rede secundária em duas camadas distintas: US de primeira classe (US_1) e US de segunda classe (US_2), os quais têm maior e menor prioridade de acesso aos recursos da RRC, respectivamente, mas mantém o UP como sendo o usuário de maior prioridade, acima do US_1 e US_2. Diante do modelo, as probabilidades de bloqueio e terminação forçada dos usuários secundários são analisadas, bem como a influência da priorização de tráfego e da configuração de largura de banda das diferentes classes no desempenho da RRC.

Este artigo encontra-se assim organizado: A Seção 2 apresenta os trabalhos relacionados. A Seção 3 apresenta o modelo proposto e a formulação das probabilidades de bloqueio e terminação forçada dos USs. A validação do modelo e a análise do impacto da priorização e configuração de largura de banda dos usuários no desempenho da RRC são apresentadas na Seção 4. A Seção 5 conclui este artigo.

2. Trabalhos Relacionados

Diferentes modelos têm sido propostos na literatura visando analisar a qualidade de serviço em RRCs. No entanto, eles apresentam limitações quanto à flexibilidade na escolha das larguras de banda, tanto de UPs quanto de USs. Em modelos (RRCs) com duas classes de serviço, a rede primária possui maior prioridade de acesso aos recursos que a rede secundária, os autores normalmente consideram que a largura de banda dos UPs é maior que a dos USs [Jiao et al. 2014; Chu et al. 2014; Chu et al. 2015]. Contudo, há também aqueles que consideram o oposto, isto é, a largura de banda dos USs sendo sempre maior ou igual a dos UPs [Jiao et al. 2010; Jiao et al. 2012; Li et al. 2012].

No modelo desenvolvido por [Zhu et al. 2007] há duas camadas de usuários (UPs e USs) na RRC, que possuem a mesma largura de banda definida por uma unidade de canal, o que torna o modelo muito simplificado. Além disso, tal trabalho apresentou falhas tanto na descrição de suas transições, que foram evidenciadas pelos autores [Ahmed et al., 2009] e [Martinez et al. 2010], quanto na validação, pois não foi adotado um modelo de simulação ou outro mecanismo para isso. Diferentemente de [Zhu et al. 2007], os autores em [Jiao et al. 2010; [Jiao et al. 2012; Li et al. 2012] consideram que os usuários podem ter larguras de banda diferentes de uma unidade de canal, mas sempre obedecendo a regra de que os USs devem ter largura de banda maior que os UPs. Já em [Jiao et al. 2014] os autores também permitem que os usuários possuam larguras de banda variadas,

porém, com a obrigatoriedade de que os UPs tenham largura de banda maior ou igual aos USs.

Em termos de restrições a largura de banda, [Jiao et al. 2014] se assemelha aos trabalhos [Chu et al. 2014; Chu et al. 2015]. No entanto, estes últimos diferem dos demais, pois propõem um modelo com três níveis de usuários ao invés de dois, i.e., a rede secundária é dividida em dois tipos de USs que podem possuir requisitos de banda diferentes, caracterizando um sistema secundário heterogêneo. Similarmente, este artigo também propõe um modelo em três camadas. Entretanto, o nosso modelo é mais flexível em termos de escolha da largura de banda que os demais, pois permite tanto que os UPs tenham maior largura de banda que os USs, quanto o contrário, o que aumenta a possibilidade de cenários que podem ser analisados. Desta forma, o modelo proposto endereça todas as configurações de largura de banda em RRCs com três níveis de prioridade, ao contrário dos demais trabalhos relacionados, e sua validação se dá através de um modelo de simulação. Além disso, nota-se que tanto [Zhu et al. 2007], quanto [Chu et al. 2014] e [CHU et al. 2015] possuem erros em algumas transições, conforme descrito na Seção "Apêndice" deste artigo.

3. Modelo Proposto

O modelo proposto adota cadeias de Markov de tempo continuo (CMTC) para representar as interações entre os UPs e USs na RRC. Nesse modelo, três tipos de usuários (UP, US_1 e US_2) compartilham N canais com as chegadas de usuários sendo modeladas como processos de Poisson, com taxas λ_{UP}, λ_{US_1} e λ_{US_2} e os tempos de serviço dos usuários são exponencialmente distribuídos com taxas μ_{UP}, μ_{US_1} e μ_{US_2}, para os UPs, US_1s e US_2s, respectivamente [Chu et al. 2014]. O modelo permite que a taxa de serviço varie de acordo com a largura de banda requerida, ou seja, quando M canais são agregados, então a taxa de serviço alcançada é $M * \mu$, o que diminui a duração de serviço do usuário.

A RRC proposta utiliza uma abordagem *overlay* centralizada que necessita de um canal de controle comum para mapear o status dos recursos ao longo da operação. Similarmente à maioria dos estudos na área, este trabalho não leva em consideração a sobrecarga imposta pelo atraso de detecção do espectro e o atraso de colisão do sistema, uma vez que são instantes curtos comparados à duração da transmissão. Em outras palavras, o módulo de controle central é responsável por escalar a fila de serviços de modo a não gerar atrasos devido a colisão, retirada ou acesso aos recursos da rede.

O mecanismo de prioridade do modelo proposto tem a seguinte forma: Quando um novo US_1 chega na RRC, um conjunto de canais não ocupados por UPs ou US_1s é aleatoriamente alocado a um novo usuário. Caso o conjunto selecionado esteja ocupado por US_2s, tais usuários vagam o conjunto de canais e buscam outros canais disponíveis na RRC para retomarem as suas comunicações. Caso não existam canais disponíveis, os US_2s terão a comunicação terminada abruptamente. Este gerenciamento de recursos pode ser realizado por uma estação base cognitiva. De forma similar, o UP segue o mesmo procedimento em relação aos US_1 e US_2. Uma vez que o UP é admitido, ele deixa a rede somente quando seu serviço é completado, o que difere dos USs, que podem ser forçados a deixarem a RRC antes da finalização das suas transmissões.

Neste modelo, uma RRC possui N canais e pode estar em vários estados, que é representado por uma tupla (i, j, k), onde i, j e k são os números de UPs, US_1s e US_2s

presentes na RRC, respectivamente. O espaço de estados viável (Ω) é gerado segundo a Eq. 1, onde B_{UP}, B_{US_1} e B_{US_2} são os requisitos de largura de banda do UP, US_1 e US_2, respectivamente.

$$\Omega = \begin{cases} (\mathrm{i, j, k}) \mid 0 \le \mathrm{i} \le \left\lfloor N/B_{UP} \right\rfloor, 0 \le j \le \left\lfloor N/B_{US_1} \right\rfloor, 0 \le k \le \left\lfloor N/B_{US_2} \right\rfloor, \\ \text{com } (i * B_{UP} + j * B_{US_1} + k * B_{US_2}) \le N \end{cases} \quad (1)$$

As expressões nas Subseções 3.1 e 3.2 descrevem todas as possíveis transições de estado para o sistema, as quais são classificadas como chegadas (solicitações) de usuários ou finalização de serviço. As transições $\gamma_{(i,j,k)}^{(i',j',k')}$ ocorrem de um estado viável (i, j, k) para outro (i', j', k') e são divididas em casos de terminações normais e forçadas, onde a primeira indica a chegada de usuários que não implica em interrupção de outros usuários, enquanto que a última necessariamente causará uma interrupção de usuário.

3.1. Transições do Estado (i, j, k) para Outros Estados

Considerando que a quantidade de recursos disponíveis (*idle*) na RCC quando o sistema está no estado (i, j, k) é dada por $\mathrm{idle} = \mathrm{N} - (\mathrm{i} * \mathrm{B_{UP}}) - (\mathrm{j} * \mathrm{B_{US_1}}) - (\mathrm{k} * \mathrm{B_{US_2}})$, tem-se as seguintes transições no modelo.

a) Chegada (solicitação) de UP na RRC

Na chegada de um UP, três situações de admissão podem ocorrer dependendo da ocupação da RRC, as quais são descritas a seguir:

- Admissão do UP sem terminação forçada de USs: caso a quantidade de recursos disponíveis seja maior ou igual à largura de banda solicitada pelo UP, ou seja, $\mathrm{B_{UP}} \le idle$, então $\mathrm{B_{UP}}$ canais serão alocados ao UP e nenhum outro usuário terá o serviço finalizado forçadamente, situação denotada pela transição da Eq. 2.

$$\gamma_{(i,j,k)}^{(i+1,j,k)} = \lambda_{UP} \quad (2)$$

- Admissão do UP e terminação forçada de US_2: caso o número de recursos disponíveis seja menor do que a largura banda do requerida pelo UP, mas a soma dos recursos disponíveis com os recursos ocupados pelos US_2s seja maior do que o solicitado pelo UP, isto é, $idle \le B_{UP} \le (idle + k * B_{US_2})$, então $\mathrm{B_{UP}}$ canais serão atribuídos ao UP e $\mathrm{Z} = \left\lceil (\mathrm{B_{UP}} - \mathrm{idle}) / \mathrm{B_{US_2}} \right\rceil$ US_2s terão seus serviços descontinuados. A Eq.3 apresenta a transição correspondente a esta situação.

$$\gamma_{(i,j,k)}^{(i+1,j,k-z)} = \lambda_{UP} \quad (3)$$

- Admissão do UP e terminação forçada de US_1 e US_2: quando a soma dos canais disponíveis com aqueles ocupados pelos US_2s não é suficiente para acomodar o novo UP, mas tal montante adicionado aos recursos utilizados pelos US_1s é, ou seja, $(idle + k * B_{US_2}) < \mathrm{B_{UP}} \le (idle + j * B_{US_1} + k * B_{US_2})$, $\mathrm{B_{UP}}$ canais são alocados

ao UP e $y = [((i * B_{UP} + j * B_{US_1}) - N + B_{UP}) / B_{US_1}]$ US_1s e k US_2 terão as comunicações terminadas abruptamente, conforme ilustra a transição na Eq.4.

$$\gamma_{(i,j,k)}^{(i+1,j-y,0)} = \lambda_{UP} \tag{4}$$

b) Chegada (solicitação) de US_1 na RRC

A chegada de um US_1 na RRC pode desencadear duas situações possíveis de admissão.

- Admissão do US_1 sem terminação forçada de US_2: caso o número de canais disponíveis seja maior ou igual a largura de banda requisitada pelo US1, isto é, $B_{US_1} \leq idle$, então B_{US_1} canais serão alocados ao US_1 sem que nenhum US_2 seja forçado a terminar a comunicação (ver Eq. 5).

$$\gamma_{(i,j,k)}^{(i,j+1,k)} = \lambda_{US_1} \tag{5}$$

- Admissão do US_1 com terminação forçada de US_2: quando a largura de banda demandada pelo US_1 não pode ser satisfeita apenas com os canais disponíveis, mas pode ser atendida considerando também os recursos utilizados pelos US_2s, ou seja, $idle < B_{US_1} \leq (idle + k * B_{US_2})$, B_{US_1} canais são designados ao novo US_1 e as comunicações de $z = [(B_{US_1} - idle) / B_{US_2}]$ US_2s são terminadas.

c) Chegada (solicitação) de US_2 na RRC

Havendo recursos disponíveis para atender à demanda do US_2 que está chegando na RRC, ou seja, $B_{US_2} \leq idle$, tal usuário é admitido e B_{US_2} canais são alocados a ele. Esta situação é representada pela transição descrita na Eq. 6.

$$\gamma_{(i,j,k)}^{(i,j,k+1)} = \lambda_{US_2} \tag{6}$$

d) Completude do Serviço do UP e dos USs

Quando os usuários têm os seus serviços completados, os recursos tornam-se disponíveis para uso pelos outros usuários da RRC. As transições que denotam a completude do serviço do UP, US1 e US2 são descritas nas Eqs. 7, 8 e 9, respectivamente.

$$\gamma_{(i,j,k)}^{(i-1,j,k)} = i * \mu_{UP} \tag{7}$$

$$\gamma_{(i,j,k)}^{(i,j-1,k)} = j * \mu_{US_1} \tag{8}$$

$$\gamma_{(i,j,k)}^{(i,j,k-1)} = k * \mu_{US_2} \tag{9}$$

3.2. Transições de outros Estados para o Estado (i, j, k)

Considerando a função $free(i,j,k) = N - (i * B_{UP}) - (j * B_{US_1}) - (k * B_{US_2})$, que denota o número de recursos disponíveis quando a RRC está no estado (i, j, k), as transições do modelo proposto são como segue.

a) Chegada (solicitação) de UP na RRC

A chegada do UP na RRC pode desencadear três situações de admissão, as quais são descritas a seguir:

- Admissão do UP sem terminação forçada de USs: se o número de canais disponíveis é maior ou igual à largura de banda solicitada pelo UP, ou seja, $B_{UP} \leq free(i-1, j, k)$, então B_{UP} canais serão alocados ao UP e nenhum outro usuário será finalizado forçadamente, o que é denotado pela transição da Eq. 10.

$$\gamma_{(i-1,j,k)}^{(i,j,k)} = \lambda_{UP} \tag{10}$$

- Admissão do UP e terminação forçada de US_2: se a quantidade de recursos disponíveis é menor do que a largura banda requerida pelo UP, ou seja, $B_{UP} > free(i-1, j, k+z')$, $k > 0$, então z' US_2s serão descartados da RRC, onde $z' = \lceil B_{UP} / B_{US_2} \rceil$ US_2s serão removidos da RRC, caso o espaço liberado seja o suficiente para o UP recém-chegado. A transição que descreve esta situação é dada na Eq. 11.

$$\gamma_{(i-1,j,k+z')}^{(i,j,k)} = \lambda_{UP} \tag{11}$$

- Admissão do UP e terminação forçada de US_1 e/ou US_2: quando a quantidade de recursos livre é menor do que a largura de banda solicitada pelo UP, ou seja, $B_{UP} > free(i-1, j, k+z')$ e $k = 0$, quatro situações podem ocorrer, com transição definida na Eq. 12.

  i. $B_{US_1} \leq B_{UP} < B_{US_2}$: se $k + z' > 0$, então somente um US_2 será descartado da RRC, ou seja, $z' = 1$. Caso $k + z' = 0$, então não existem US_2s a serem descartados, assim $y' = \lceil B_{UP} / B_{US_1} \rceil$ US_1s terão as comunicações finalizadas forçadamente.

  ii. $B_{UP} < B_{US_1}$ e $B_{UP} < B_{US_2}$: se $k + z' > 0$, então somente um US_2 será descartado, ou seja, $z' = 1$ e $y' = 0$. Caso $k + z' = 0$, então apenas um US_1 será descartado, ou seja, $z' = 0$ e $y' = 1$.

  iii. $B_{UP} \geq B_{US_1}$ e $B_{UP} \geq B_{US_2}$: neste caso, y' pode variar de 0 a B_{UP} / B_{US_1}, enquanto z' varia de 0 a B_{UP} / B_{US_2}, dado que $y' * B_{US_1} * z' B_{US_2} = (B_{UP} - free(i-1, j+y', k+z'))$.

  iv. $B_{US_2} \leq B_{UP} < B_{US_1}$: se $B_{UP} \leq (free(i-1, j+y', k+z') + (k+z') * B_{US_2})$, então z' varia de 0 a B_{UP} / B_{US_2} dado que $z' * B_{US_2} = (B_{UP} - free(i-1, j+y', k+z')$ e $y' = 0$. Caso $z' * B_{US_2} > (free(i-1, j+y', k+z') + (k+z') * B_{US_2})$, não somente todos os US_2s serão descartados, mas também um US_1 ($z' = k$ e $y' = 1$) para liberar recursos para o UP que está requisitando.

$$\gamma_{(i-1,j+y',k+z')}^{(i,j,k)} = \lambda_{UP} \tag{12}$$

b) Chegada (solicitação) de US1 na RRC

A chegada de US1 na RCC pode desencadear duas situações de admissão, dependendo de como os canais estão ocupados:

- Admissão do US1 sem terminação forçada de US2: quando o montante de recursos disponíveis é maior ou igual à largura de banda requisitada pelo US1, B_{US_1} canais são alocados ao novo US1 e nenhum US2 terá a comunicação finalizada abruptamente. A Eq. 13 apresenta a transição relativa a esta situação.

$$\gamma_{(i,\mathrm{j}-1,k)}^{(i,j,k)} = \lambda_{US_1} \tag{13}$$

- Admissão do US1 com terminação forçada de US2: caso não haja recursos disponíveis para admitir um novo US1 na RRC, mas através do descarte de US2s se obtém canais suficientes para isso, ou seja, $free(i, j+y', \mathrm{k}+\mathrm{z}') + (\mathrm{k}+\mathrm{z}') * B_{US_2} \geq B_{US_1} > free(i, j-1, k+z')$, então mais de uma situação pode leva o sistema ao estado (i, j, k), com transição definida na Eq. 14.

  i. Se $\mathrm{B}_{\mathrm{US}_1} \leq B_{US_2}$ e $(k+z') > 0$, então $z' = 1$.

  ii. Se $\mathrm{B}_{\mathrm{US}_1} > B_{US_2}$, então z' pode assumir até $\lceil B_{US_1} / B_{US_2} \rceil$.

$$\gamma_{(i,\mathrm{j}-1,k+z')}^{(i,j,k)} = \lambda_{US_1} \tag{14}$$

c) Chegada (solicitação) de US2 na RRC

Caso haja canais disponíveis suficientes para atender a demanda do novo US2, ou seja, $B_{US_2} \leq free(i, j, k-1)$, então B_{US_2} canais serão alocados a ele, e a transição do sistema é dada pela Eq.15.

$$\gamma_{(i,\mathrm{j},k-1)}^{(i,j,k)} = \lambda_{US_2} \tag{14}$$

d) Finalização do Serviço do UP e dos USs

As transições do modelo que denotam a finalização (normal) da comunicação do UP, US1 e US2 e conduzem o sistema ao estado (i, j, k) são dadas nas Eqs. 15,16 e 17, respectivamente.

$$\gamma_{(i+1,\mathrm{j},k)}^{(i,j,k)} = (i+1) * \mu_{UP} \tag{15}$$

$$\gamma_{(i,\mathrm{j}+1,k)}^{(i,j,k)} = (j+1) * \mu_{US_1} \tag{16}$$

$$\gamma_{(i,\mathrm{j},k+1)}^{(i,j,k)} = (k+1) * \mu_{US_2} \tag{17}$$

Um exemplo de diagrama de transição de estados para uma RRC com dois canais (N=2) e requisitos de largura de banda $B_{UP} = 1, B_{US_1} = 1 \mathrm{e}\ B_{US_2} = 1$ é ilustrado na Fig. 1, onde os fluxos horizontais para a direita (esquerda) representam a chegada (partida) de

US_1s, os fluxos verticais para cima (baixo) denotam a chegada (partida) de UPs e os fluxos diagonais para dentro (fora) do plano indicam a chegada (partida) de US_2s.

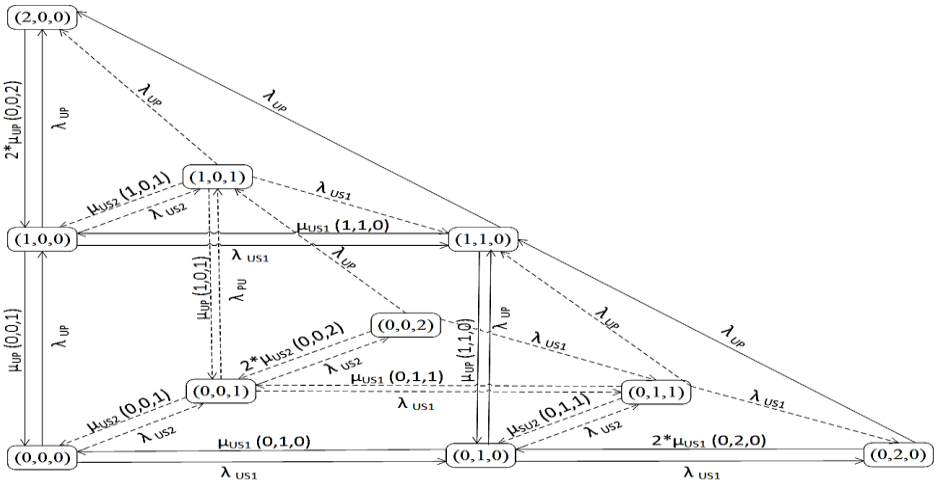


Figura 1. Diagrama de transição de estados com $N = 2$ **e** $B_{UP} = B_{US_1} = B_{US_2} = 1$

Para obter as probabilidades de estado estacionário do modelo, o sistema linear formado pelas equações de balanceamento (Eq. 18) e condição de normalização (Eq.19) deve ser resolvido, onde $\pi(i, j, k)$ denota a probabilidade de estado estacionário para o estado (i, j, k) e $I(i, j, k)$ é a função de indicação de estado viável (Eq. 20).

$$\sum_{i'=0}^{\left\lfloor \frac{N}{B_{UP}} \right\rfloor} \sum_{j'=0}^{\left\lfloor \frac{N}{B_{US_1}} \right\rfloor} \sum_{k'=0}^{\left\lfloor \frac{N}{B_{US_2}} \right\rfloor} p(i,j,k) * g_{(i,j,k)}^{(i',j',k')} * I(i,j,k) * I(i',j',k') = \sum_{i'=0}^{\left\lfloor \frac{N}{B_{UP}} \right\rfloor} \sum_{j'=0}^{\left\lfloor \frac{N}{B_{US_1}} \right\rfloor} \sum_{k'=0}^{\left\lfloor \frac{N}{B_{US_2}} \right\rfloor} p(i',j',k') * g_{(i',j',k')}^{(i,j,k)} * I(i,j,k) * I(i',j',k'), \; onde \; (i,j,k) \neq (i',j',k') \tag{18}$$

$$\sum_{i=0}^{\left\lfloor \frac{N}{B_{UP}} \right\rfloor} \sum_{j=0}^{\left\lfloor \frac{N}{B_{US_1}} \right\rfloor} \sum_{k=0}^{\left\lfloor \frac{N}{B_{US_2}} \right\rfloor} p(i,j,k) * I(i,j,k) = 1 \tag{19}$$

$$I(i,j,k) = \begin{cases} 1, \; se \; (i,j,k) \in \Omega \\ 0, \; caso \; contrário \end{cases} \tag{20}$$

A solução do sistema linear formado pelas Eqs. 19 e 20 é o vetor de estado estacionário. Ele é usado para formular as probabilidades de bloqueio e terminação forçada do US, que são métricas importantes na avaliação de desempenho da comunicação secundária na RRC.

3.3. Probabilidade de Bloqueio de US

O US é bloqueado quando ele tenta acessar a RRC, mas não existem recursos suficientes. Assim, a probabilidade de bloqueio é a porcentagem de USs que não são aceitos pela RRC, sendo um indicador útil de QoS. As probabilidades de bloqueio do US_1 e US_2,

denotadas por PB_{US_1} (ver Eq. 21) e PB_{US_2} (ver Eq. 22), respectivamente, são dadas pela soma das probabilidades de estado estacionário dos estados que caracterizam a rede totalmente ocupada para cada tipo de usuário.

$$PB_{US_1} = \sum_{i=0}^{\left\lfloor \frac{N}{B_{UP}} \right\rfloor} \sum_{j=\left\lceil \frac{N-(i*B_{UP})}{B_{US_1}} \right\rceil}^{\left\lfloor \frac{N}{B_{US_1}} \right\rfloor} \sum_{k=0}^{\left\lfloor \frac{N}{B_{US_2}} \right\rfloor} \pi(i,j,k) * I(i,j,k) \tag{21}$$

$$PB_{US_1} = \sum_{i=0}^{\left\lfloor \frac{N}{B_{UP}} \right\rfloor} \sum_{j=\left\lceil \frac{N-(i*B_{UP})}{B_{US_1}} \right\rceil}^{\left\lfloor \frac{N}{B_{US_1}} \right\rfloor} \sum_{k=\left\lceil \frac{N-(i*B_{UP})-(j*B_{US_1})}{B_{US_2}} \right\rceil}^{\left\lfloor \frac{N}{B_{US_2}} \right\rfloor} \pi(i,j,k) * I(i,j,k) \tag{22}$$

3.4. Probabilidade de Terminação Forçada de US

Uma vez que os USs são admitidos na RRC, a comunicação deles pode ser terminada abruptamente pela chegada de usuários de maior prioridade, causando a degradação da comunicação secundária. O US_1 sofre terminação forçada quando não existem recursos para atender o UP que está chegando mesmo considerando os canais adotados pelos US_2s, ou seja, $[idle + (k * B_{US_2})] < B_{UP}$. Dado que o estado do sistema é (i,j,k) e usando $idle$ para denotar o número de recursos disponíveis na RRC quando ela está no estado (i,j,k), o número de US_1s terminados forçadamente pela chegada de um UP é $y = \left\lceil ((i * B_{UP}) + (j * B_{US_1}) - N + B_{UP}) / B_{US_1} \right\rceil$. Realizando o produto entre y e a taxa de UPs que ocasionam terminação forçada de US_1s, tem-se a taxa de US1s terminados forçadamente na RRC, a qual é apresentada no numerador da Eq. 23. Dividindo a taxa de US1s descartados pela taxa de US1s admitidos na RRC, tem-se a probabilidade de terminação forçada do US_1, a qual dada pela Eq. 23.

$$PTF_{US_1} = \frac{\lambda_{UP}}{(1 - PB_{US_1}) * \lambda_{US_1}} \sum_{i=0}^{\left\lfloor \frac{N}{B_{UP}} \right\rfloor} \sum_{j=0}^{\left\lfloor \frac{N}{B_{US_1}} \right\rfloor} \sum_{k=0}^{\left\lfloor \frac{N}{B_{US_2}} \right\rfloor} \pi(i,j,k) * y(i,j,k) * I(i,j,k) * y(i+1, j-y, k) \tag{23}$$

O US de mais baixa prioridade (US_2) está sujeito a eventos de terminação forçada nos dois casos seguintes. Na chegada de UP, quando não existem recursos disponíveis para atender o novo UP, ou seja, $B_{UP} > idle$, e, da mesma forma, na chegada de um US_1 com $B_{US_1} > idle$. O número de US_2s descartados por cada usuário de maior prioridade é $z_1 = \left\lceil (B_{UP} - idle) / B_{US_2} \right\rceil$ no primeiro caso e $z_2 = \left\lceil (B_{US_1} - idle) / B_{US_2} \right\rceil$ no último. Assim, as taxas de US_2s descartados no primeiro (C1) e no segundo (C2) caso são dadas pelas Eq. 24 e 25, respectivamente. Deste modo, a probabilidade de terminação forçada de US_2s é obtida através da razão entre a taxa total de US_2s descartados e a taxa de US_2s admitidos na RRC, dada na Eq. 26.

$$C_1 = \sum_{i=0}^{\left\lfloor \frac{N}{B_{UP}} \right\rfloor} \sum_{j=0}^{\left\lfloor \frac{N}{B_{US_1}} \right\rfloor} \sum_{k=0}^{\left\lfloor \frac{N}{B_{US_2}} \right\rfloor} \lambda_{UP} * \pi(i,j,k) * z_1 * I(i,j,k) * y(i+1, j, k-z) \tag{24}$$

$$C_2 = \sum_{i=0}^{\left\lfloor \frac{N}{B_{UP}} \right\rfloor} \sum_{j=0}^{\left\lfloor \frac{N}{B_{US_1}} \right\rfloor} \sum_{k=0}^{\left\lfloor \frac{N}{B_{US_2}} \right\rfloor} \lambda_{US_1} * \pi(i, j, k) * z_2 * I(i, j, k) * y(i, j+1, k-z) \quad (25)$$

$$PTF_{US_2} = \frac{C_1 + C_2}{(1 - PB_{US_2}) * \lambda_{US_2}} \quad (26)$$

4. Validação do Modelo e Análise de Resultados

Para finalidade de validação, os resultados obtidos através do modelo analítico (linhas sólidas) foram comparados ao modelo de simulação (marcadores) baseado em eventos discretos, desenvolvido em MATLAB. Cada evento ocorre em um instante de tempo único, podendo causar uma mudança de estado do sistema. Ao contrário das simulações baseadas em tempo contínuo, não ocorre mudança no estado entre os eventos, logo, pode-se assumir que assim que processada uma requisição, o sistema passe a consumir o próximo evento, que é definido a partir da ordem de chegada, i.e., a disciplina de filas adotada é FIFO (*First In, First Out*). Como a criação dos eventos é realizada de forma independente, o passo final é unir as três filas de eventos geradas (uma para cada tipo de usuário). Diferentemente do modelo analítico, neste tipo de simulação as métricas não são derivadas, são médias de diferentes execuções.

Foi considerado um cenário composto de 4 canais de comunicação ($N = 4$), taxas de chegada de US_1 e US_2 dadas por $\lambda_{US_1} = 1$ e $\lambda_{US_2} = 1$ usuários por unidades de tempo e taxas de serviço de UP, US_1e US_2 iguais a $\mu_{UP} = 1$, $\mu_{US_1} = 1$e $\mu_{US_2} = 1$ usuários por unidade de tempo, respectivamente. Embora se tenha adotado 4 canais, qualquer quantidade de canais pode ser considerada no modelo proposto, com o requisito que N deve ser maior ou igual ao maior requisito de largura de banda dos usuários. Os resultados do modelo de simulação foram obtidos através de 100 instâncias de execução com tempo de simulação igual a 10000 unidades de tempo. Os resultados médios são apresentados considerando um nível de confiança de 95%.

Para avaliar o desempenho da RRC sob diferentes níveis de ocupação primária (carga de UP), a taxa de chegada do UP foi variada. Trabalhos anteriores têm limitação nas configurações de largura de banda de cada tipo de usuário, ao passo que no modelo proposto não há esta limitação. Deste modo, seis cenários, que representam as possíveis configurações (relações de desigualdade) de largura de banda entre os usuários foram considerados, o que permite avaliar o impacto da configuração de largura de banda na comunicação dos USs. A Fig. 2 ilustra as seis configurações, onde, neste trabalho, os valores para as larguras de banda dos usuários foram definidos como 1, 2 e 4 canais. O desempenho da comunicação secundária foi avaliado em termos das probabilidades de bloqueio e terminação forçada dos USs.

A Fig. 3 apresenta os resultados da probabilidade de bloqueio para o US_1. Nota-se que a largura de banda requerida impacta significamemte no nível de admissão do US_1 na RRC. Por exemplo, requisitar uma largura de banda maior para transmissão pode ser difícil de ser atendida pela RRC, dependendo do tamanho da banda requisitada e do nível de ocupação primária, o que afeta a admissão de US_1s na RRC. Por esta razão, as configurações 3 e 4 (com marcadores “x” e “o”, respectivamente), que atribuíram as maiores larguras de banda aos US1s, apresentaram as maiores probabilidades de bloqueio

de US_1s. O mesmo se aplica aos resultados de probabilidade de bloqueio de US_2s ilustrados na Fig. 4, onde as configurações 5 e 6 (com marcadores "+" e "□"), em que o US_2 requer maior largura de banda para realizar a sua comunicação, apresentam as maiores probabilidades de bloqueio do US_2.

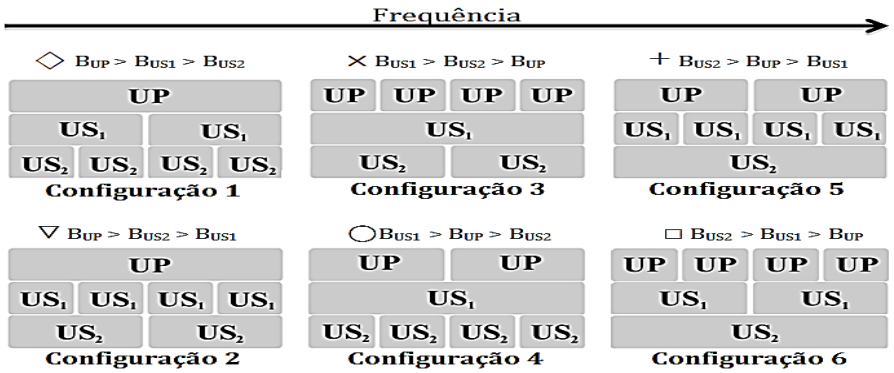


Figura 2. Configurações de largura de banda endereçadas pelo modelo

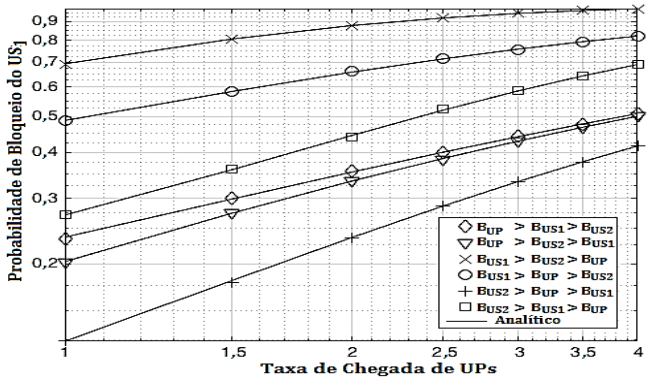


Figura 3. Resultados da Probabilidade de Bloqueio do US_1

Por outro lado, elevado requisito de largura de banda diminui a duração de serviço do usuário. Assim, quando o UP adota mais canais na sua comunicação, ele libera os recursos mais rapidamente, proporcionando a redução no bloqueio de US_1 e US_2. Isto justifica a diferença de desempenho entre as configurações 3 (maior probabilidade de bloqueio) e 4 na Fig. 3, e as configurações 5 e 6 (maior probabilidade de bloqueio) na Fig. 4, quando o requisito de largura de banda do UP varia.

Os melhores pares de resultados de probabilidade de bloqueio na rede secundária foram obtidos pelas configurações 2 e 5 (ver Fig. 3) e 1 e 4 (ver Fig. 4). Isto ocorreu, pois quanto menor é a largura de banda requisitada pelo US, mais facilmente ele pode ser aceito na RRC, mesmo em condições de alta carga na RRC. Neste aspecto, nota-se que, embora a priorização exerça um papel significativo na RRC, a configuração de largura de banda adotada na rede impacta no seu desempenho. Por exemplo, na configuração 4 com um cenário de alta carga primária ($\lambda_{UP} = 4$), a probabilidade de bloqueio do US_2 (ver Fig. 4) é menor (42,5%) do que a correspondente probabilidade de bloqueio do US_1 apresentada na Fig. 3 (50%) para um cenário esparso ($\lambda_{UP} = 1$), embora o US_1 tenha maior prioridade de acesso do que o US_2.

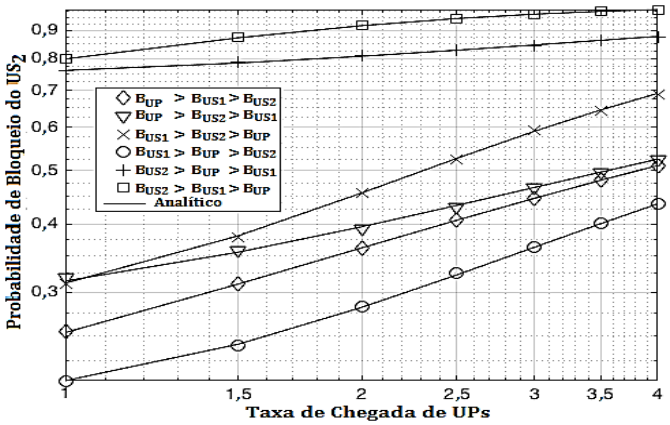


Figura 4. Resultados da Probabilidade de Bloqueio do US_2

Os resultados em termos de probabilidade de terminação forçada de US_1 e US_2 são apresentados na Figs. 5 e 6, respectivamente. Nota-se que quando os USs adotam largura de banda menor, tais como os definidos na configuração 2 (marcador "∇") para o US_1 (ver Fig. 5) e configuração 1 (marcador "◊") para o US_2 (ver Fig. 6), a chance de suas comunicações serem terminadas forçadamente aumenta, pois com menor largura de banda, maior é tempo de serviço e a chance de chegada de usuários de maior prioridade na RRC. De forma similar, a adoção de configurações com maior largura de banda na comunicação secundária também não é efetiva em termos de probabilidade de terminação forçada, pois, embora uma duração de serviço menor seja alcançada, a possibilidade da chegada de um usuário de maior prioridade requerer os canais adotados pela comunicação secundária pode aumentar, forçando o US a liberar os canais em uso e buscar outros recursos disponíveis para retomar a sua comunicação. Entretanto, pode ser muito difícil encontrar grandes montantes de recursos disponíveis.

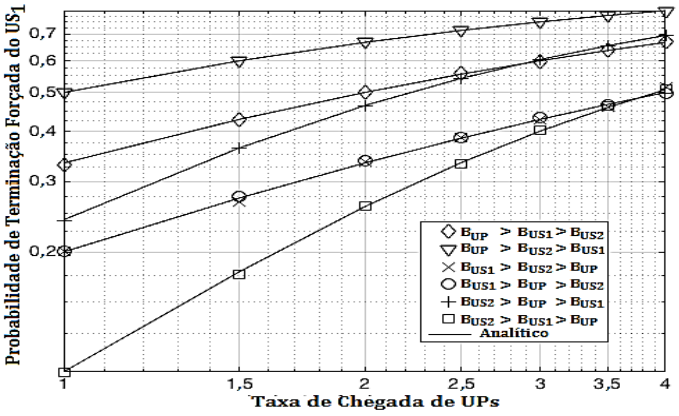


Figura 5. Resultados da Probabilidade de Terminação Forçada do US_1

Configurações com largura de banda intermediária, tais como as configurações 6 (marcador "□") na Fig. 5 para o US_1 e 3 (marcador "x") na Fig. 6 para o US_2, alcançaram o melhor compromisso entre o montante de recursos ocupados e o tempo de serviço total, o que mitigou a probabilidade de terminação forçada do US_1 e US_2, respectivamente.

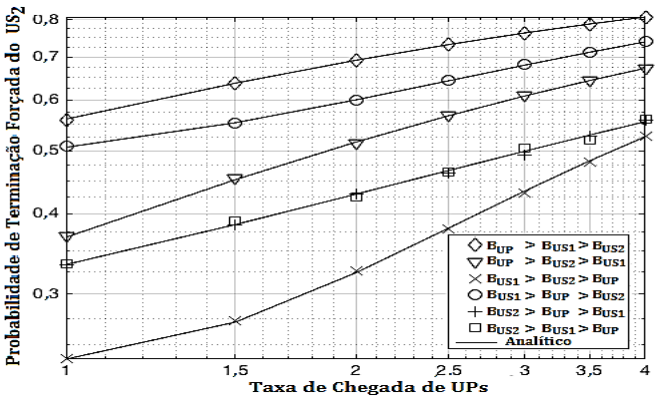


Figura 6. Resultados da Probabilidade de Terminação Forçada do US_2

5. Conclusão

Este trabalho propôs e validou um modelo baseado em CMTC e flexível em termos de largura de banda para analisar RRCs com ambiente secundário heterogêneo e sua posterior validação. Conduzimos uma análise formal do desempenho da comunicação secundária, discorrendo sobre diversos cenários (configurações de largura de banda entre classes de usuários) e níveis de carga do UP. Tal exame gerou a apresentação ponderada de uma série de ideias úteis acerca de como diferentes aplicações (com requisitos específicos de largura de banda) podem ser estruturadas na RRC buscando prover bons níveis de QoS à comunicação secundária (em termos das probabilidades de bloqueio e terminação forçada dos USs). Os resultados ilustraram como a configuração de largura de banda e o mecanismo de priorização impactam na comunicação secundária em uma CRN com três camadas sob diferentes cargas de UPs.

O modelo proposto é o mais flexível da literatura atual, uma vez que os trabalhos anteriores não endereçam todas as possibilidades de configurações de largura de banda para os usuários em um único modelo com três camadas de usuários. Consideramos este modelo como um passo importante em direção a um modelo generalista, capaz de suportar um número genérico de classes de prioridades e técnicas de adaptação de recursos, tais como reserva de canais e agregação de portadoras. Consubstanciados nesses resultados, podemos afirmar que seria possível reproduzir o comportamento de padrões como o IEEE 802.11p – que possui quatro categorias de tráfego, separadas por prioridade além de requerer diferentes larguras de banda para cada uma delas (e.g. *Beacons* de sinalização e *streaming* de vídeo).

Referências

Ahmed, W., Gao, J., Suraweera, H. and Faulkner, M. (2009) “Comments on "Analysis of cognitive radio spectrum access with optimal channel reservation”, IEEE Transactions on Wireless Communications, v. 8, n. 9, p. 4488-4491, 2009.

Chu, T., Phan, H. and Zepernick, H. (2014) "Dynamic Spectrum Access for Cognitive Radio Networks With Prioritized Traffics", IEEE Communications Letters, vol. 18, no. 7, pp. 1218-1221, 2014.

Chu, T., Phan, H. and Zepernick, H. (2015) "Channel Reservation for Dynamic Spectrum Access for Cognitive Radio Networks With Prioritized Traffic" in IEEE International Conference on Communication Workshop (ICCW), 2015.

Fitch, M., Nekovee, M., Kawade, S., Briggs, K. and Mackenzie, R. (2011), "Wireless Service Provision in TV White Space with Cognitive Radio Technology: A Telecom Operator's Perspective and Experience", IEEE Communication Magazine, vol. 49, pp. 64-73, 2011.

Jiao, L., Pla, V. and Li, F. (2010) "Analysis on channel bonding/aggregation for multi-channel cognitive radio networks", European Wireless Conference (EW), 2010.

Jiao, L., Li, F. and Pla, V. (2012) "Modeling and Performance Analysis of Channel Assembling in Multichannel Cognitive Radio Networks with Spectrum Adaptation", IEEE Trans. Veh. Technol., v. 61, n. 6, p. 2686-2697, 2012.

Jiao, L., Balapuwaduge, I. A. M., Li, F. Y., Pla, V. (2014) "On the Performance of Channel Assembling and Fragmentation in Cognitive Radio Networks," IEEE Transactions on Wireless Communications, vol.13, no.10, pp.5661-5675, 2014.

Li, L., Zhang, S., Wang, K. and Zhou, W. (2012) "Combined Channel Aggregation and Fragmentation Strategy in Cognitive Radio Networks," arxiv preprint: http://arxiv.org/pdf/1203.4913v2.pdf, 2012.

Liang, Y-C., Chen, K-C., Li, G.Y.; Mahonen, P. (2011) "Cognitive radio networking and communications: an overview", IEEE Transactions on Vehicular Technology, vol.60, no.7, pp.3386-3407, 2011.

Martinez-bauset, J., Pla, V. and Pacheco-paramo, D. (2009) "Comments on "analysis of cognitive radio spectrum access with optimal channel reservation". IEEE Communications Letters, vol. 13, n. 10, 2009.

Wang, J., Gosh, M. and Challapali, K. (2011) "Emerging Cognitive Radio Applications: A Survey", IEEE Communications Magazine, vol 49, pp. 74-81, 2011.

Zhu, X., Shen, L.; Yum, T. (2007) "Analysis of Cognitive Radio Spectrum Access with Optimal Channel Reservation", IEEE Communications Letters, v. 11, n. 4, p. 304-306, 2007.

Apêndice: Erro na Descrição de Transição em [CHU et al. 2014] e [Chu et al. 2015]

Estes artigos apresentam o mesmo erro na transição que adota o número de usuários secundários de maior prioridade (US_1) a serem removidos da rede devido a uma chegada de UP. O erro encontra-se na fórmula para calcular o número de US_1s a serem removidos (j'), dada por $0 \leq j' \leq \lceil (N*(M-i) - j*N_1) / N_1 \rceil$, onde $M*N$ é o número total de canais, N_1 é o número de canais que os SU_1s ocupam e $\lceil . \rceil$ é a função teto. Caso utilizemos o exemplo $M = 2$, $N = 1$, $N_1 = 1$ (válido segundo os autores), a transição do estado $(1,1,0)$ para o estado $(2,0,0)$ não irá ocorrer, pois diferentemente do esperado onde o cálculo de j' deveria resultar em '1', pela função descrita teremos $0 \leq j' \leq \lceil (1*(2-2) - 0*1) / 1 \rceil$ que resulta em 0, tornando a transição inválida para este exemplo.

Um Protocolo de Alocação Dinâmica de Canais para Ambientes Médicos sob Múltiplas Estações Base

Bruno M. Cremonezi[1], **Alex B. Vieira**[1], **Michele Nogueira** [2], **José A. M. Nacif** [3]

[1]Depto. de Ciência da Computação – Universidade Federal de Juiz de Fora (UFJF)
[2]Depto. de Informática – Universidade Federal do Paraná (UFPR)
[3]Instituto de Ciências Exatas e Tecnológicas – Universidade Federal de Viçosa (UFV)

brunomarques@ice.ufjf.br, alex.borges@ufjf.edu.br
michele@inf.ufpr.br, jnacif@ufv.br

Abstract. *With estimates predicting a growth up to 3 billion wearable sensors by 2025, health monitoring over wireless networks is at the same time increasingly popular and a challenging task. Prone to high error rates and interferences, wireless transmission is vulnerable to high latency and low throughput, being a serious risk to users. In general, existing solutions have addressed interferences in both wireless body and wireless local area networks. However, they rarely cope with the problem taking directly into account the minimum requirements for medical applications. This work presents PDAC, a medical application aware MAC-layer* **P***rotocol for* **D***ynamic Channel* **A***llo***C***ation. PDAC offers a solution to reduce interferences and allows the cooperation of multiple base stations in dense medical-hospital environments. When using PDAC, base stations can work collaboratively to reach the requirements for medical applications. Simulation results indicate that, in a realistic medical environment, PDAC is able, in average, to increase throughput by 30% and reduce latency by 40%, when compared to state-of-the-art frequency allocation protocols.*

Resumo. *Com estimativas de crescimento de até 3 bilhões de sensores vestíveis até 2025, o monitoramento da saúde por redes sem fio tem se tornado cada vez mais popular e desafiador. Sujeitas a altas taxas de erros e interferências, as transmissões sem fio são vulneráveis a alta latência e baixa vazão, resultando em sérios riscos para os usuários. Em geral, as soluções existentes tratam de interferências em redes corporais e redes locais sem fio. Entretanto, raramente, focam neste problema considerando os requisitos mínimos exigidos pelas aplicações médicas. Este trabalho apresenta o PDAC, um protocolo da camada MAC ciente dos requisitos das aplicações médicas promovendo a alocação dinâmica de canais em cenários hospitalares densos. O PDAC oferece uma solução para reduzir interferências e permite a cooperação de múltiplas estações base no mesmo ambiente médico-hospitalar. Ao utilizar o PDAC, as estações base trabalham de forma colaborativa para atender aos requisitos necessários para as aplicações médicas. Os resultados de simulação indicam que, em um ambiente médico realista, o PDAC é capaz de, em média, aumentar a vazão em 30% e reduzir a latência em 40% quando comparado com protocolos de alocação de frequência do estado da arte.*

1. Introdução

Os sistemas de saúde vêm se beneficiando cada vez mais da evolução das tecnologias de comunicação sem fio [Omeni et al. 2008]. Estimativas preveem que a população mun-

dial utilizará até 2025 em torno de 3 bilhões de sensores vestíveis com capacidade de comunicação, como as redes corporais e as redes locais sem fio. As redes corporais e as redes locais sem fio oferecem a transmissão dos dados monitorados por sensores incorporados sob ou sobre a pele dos usuários [Movassaghi et al. 2014]. Tipicamente, as redes corporais transmitem os dados a um nó coordenador (ex. um *smartphone*) que, por sua vez, envia os dados para um repositório central ou nuvem, onde são finalmente armazenados e analisados. Tais análises podem resultar em alertas que auxiliam no atendimento emergencial de pacientes, ex. no caso de um ataque cardíaco ou uma parada respiratória.

As redes sem fio apresentam diversas limitações, em geral, causadas pelas características inerentes da comunicação sem fio e/ou por utilizarem a faixa não licenciada do espectro de radiofrequência. É comum, por exemplo, que duas ou mais redes corporais diferentes utilizem simultaneamente o mesmo canal de comunicação sem fio (ou canais parcialmente sobrepostos) [Fang et al. 2010]. Isto potencializa interferências, acarretando um maior número de retransmissões e perdas de pacote na rede. Como consequência, essas redes apresentam altas latências e queda da vazão [Fang et al. 2010]. Particularmente, em um ambiente médico-hospitalar, observa-se um aumento na densidade de redes corporais e redes locais sem fio [Baker and Hoglund 2008, Fang et al. 2010]. Diante do número reduzido de canais sem sobreposição na faixa de frequência não licenciada, quanto maior a densidade, maior a latência e menor a vazão. Este cenário é especialmente desafiador em aplicações médicas, nas quais os requisitos de latência e perda de pacotes são menores que, respectivamente, 200 ms [Baker and Hoglund 2008] e 1% [Yu et al. 2006].

Para reduzir este problema, diversos trabalhos propuseram técnicas de alocação de canais entre as redes corporais co-localizadas [Doost-Mohammady and Chowdhury 2012, Phunchongharn et al. 2010, Lee et al. 2011]. Estas técnicas permitem a utilização simultânea de diferentes canais pelas redes corporais para transmissão de dados, evitando interferências. Geralmente, tais mecanismos e técnicas de alocação de canal dependem de uma entidade centralizada. Entretanto, em um ambiente médico, as abordagens centralizadas podem comprometer o desempenho da rede por sobrecarregarem a entidade centralizadora devido à alta densidade de redes corporais. Além disso, raramente as técnicas de alocação de canais são projetadas levando em conta os requisitos de latência e vazão específicos para as aplicações médicas.

Este trabalho apresenta PDAC (**P***rotocol for* **D***ynamic Channel* **A***llo***C***ation*), um protocolo da camada MAC para alocação dinâmica de canais ciente dos requisitos de aplicações médicas. O PDAC oferece uma solução para reduzir interferências entre redes corporais sem fio e permite a existência de múltiplas estações base no mesmo ambiente médico-hospitalar. Ao utilizar o PDAC, as estações base trabalham de forma colaborativa para atender aos requisitos das aplicações médicas. O protocolo PDAC é inspirado em uma solução gulosa do problema de coloração de grafos. Desta forma, assume-se um grafo de interferências entre as estações base, no qual cada vértice representa uma estação base e cada aresta representa conflitos entre os canais utilizados pelas estações base. Cada canal disponível é mapeado nas cores de um problema de coloração de grafo. De modo geral, cada estação base busca uma solução local para a alocação de canais e comunica para as estações bases conflitante o seu atual estado. Essa informação é utilizada por outras estações para auxiliar na alocação de frequências, buscando reduzir interferências.

A avaliação de desempenho do protocolo foi realizada por simulações no Castalia [Boulis et al. 2011] sob cenários realistas de um ambiente médico. Os resultados demonstram que o PDAC é capaz, em média, de aumentar a vazão da rede em 30% e reduzir a latência em 40%, quando comparado com uma abordagem de alocação de canais representativa da literatura. O PDAC, quando utilizado com múltiplas estações base, atende aos requisitos de atraso e perdas de pacotes das aplicações médicas, apresentando latência média de 120 ms e perdas inferiores a 0,01%.

Este artigo está organizado como segue. A Seção 2 discute os trabalhos relacionados. Na Seção 3, descrevemos o modelo do sistema e detalhamos o novo protocolo. A Seção 4 apresenta a avaliação de desempenho do PDAC. Finalmente, a Seção 5 discute as conclusões e direções futuras deste trabalho.

2. Trabalhos Relacionados

Em geral, os protocolos da camada MAC têm o objetivo de reduzir interferências, proporcionando menor latência e maior vazão na transmissão dos dados. Uma abordagem comum consiste no uso de múltiplos canais de transmissão [Soua and Minet 2011]. Em [Kyasanur and Vaidya 2006], os autores estudaram o uso de múltiplos canais em redes *ad hoc* sem fio. Eles investigaram o impacto do número de interfaces na capacidade da rede e concluíram que mesmo em cenários nos quais o número de interfaces é menor que o número de canais disponíveis, a rede apresenta ganhos em vazão e latência. Apesar dos benefícios no uso de múltiplos canais, o processo de alocação ainda é desafiador.

Os protocolos multicanais, em geral, seguem o comportamento do protocolo DCAA de alocação dinâmica de canais [Lee et al. 2011]. Esse protocolo oferece uma abordagem de salto de frequências para alocação de canais a uma comunicação entre estação base e cliente. A estação base continuamente altera sua sintonização entre os canais, permanecendo em um canal por um curto intervalo de tempo. O cliente que deseja iniciar uma comunicação escolhe um canal aleatório e envia uma requisição. Caso o canal esteja ocupado ou a estação base não responda, o cliente salta para o próximo canal e reinicia o processo de solicitação de comunicação. Caso a estação base responda à requisição, a transmissão dos dados é iniciada. A abordagem de Lee et al. simplifica o processo de alocação e de gerência dos canais. No entanto, quanto maior o número de canais disponíveis, menor a probabilidade de o cliente sintonizar no mesmo canal da estação base. Este problema é chamado de *surdez*, ocorrendo quando o cliente tenta se comunicar em um canal diferente do que a estação base está sintonizada. Assim, o DCAA não atende os requisitos de latência para as aplicações médicas, além de um eventual desperdício de energia durante a tentativa de pareamento.

A fim de evitar o problema de *surdez*, alguns trabalhos utilizam uma abordagem híbrida de alocação de canais [Phunchongharn et al. 2010, Doost-Mohammady and Chowdhury 2012]. As estratégias híbridas assumem que cada estação base está equipada com múltiplas interfaces de comunicação sem fio. Por exemplo, [Phunchongharn et al. 2010] apresentaram um sistema multicanais no qual um canal é dedicado ao controle e outro à transferência dos dados. Nesse sistema, a transmissão dos dados ocorre em dois estágios. O canal de controle é utilizado para o *handshaking*, no qual a frequência e a potência de transmissão são negociadas e posteriormente o cliente sintoniza um canal dedicado para a transmissão dos dados.

Em [Doost-Mohammady and Chowdhury 2012], os autores também apresentaram um protocolo seguindo a divisão em canal de controle e canal de dados. Entretanto, ao invés de um único canal de dados, o protocolo utiliza múltiplos canais de dados. Além disso, os autores ofereceram formas de evitar interferências em aparelhos médicos presentes no ambiente hospitalar e apresentaram uma diferenciação de QoS entre os clientes.

Em nenhum desses trabalhos, cenários com múltiplas estações base são considerados. As etapas de controle e transmissão de dados são conduzidas por uma única estação base, sujeita à sobrecarga em um ponto único. Entretanto, em um ambiente médico, é observada a presença de múltiplas estações base compartilhando uma mesma região geográfica [Baker and Hoglund 2008]. As estações base próximas, operando no mesmo canal, levam a um aumento no número de interferências, implicando em um aumento da latência e uma degradação na vazão. Esse problema pode ser reduzido caso as estações possuam um mecanismo de comunicação entre elas, coordenando o uso das frequências na alocação de canais aos clientes. Porém, assegurar a sincronização entre as estações base adiciona um novo nível de complexidade ao problema de alocação de frequências.

Diferente desses trabalhos prévios, este trabalho foca na transmissão dos dados considerando os requisitos das aplicações médicas. Além disso, o protocolo proposto tem como base a coordenação entre múltiplas estações base.

3. Protocolo Dinâmico de Alocação de Canais

Esta seção detalha o protocolo PDAC. A Subseção 3.1 apresenta o modelo do sistema seguido neste trabalho. A Subseção 3.2 mostra uma visão geral do protocolo onde cada de suas fases é explicada.

3.1. Modelo do Sistema

A Figura 1 ilustra a arquitetura do ambiente hospitalar considerada nesta trabalho, ressaltando os dois principais tipos de comunicação entre os dispositivos: a comunicação dentro da rede corporal sem fio (intra-WBAN) e a comunicação entre as redes corporais (inter-WBAN). Cada paciente possui um conjunto de sensores sem fio distribuídos ou implantados no seu corpo. Esses sensores monitoram a saúde do paciente coletando, por exemplo, dados sobre seus sinais vitais. Os sensores transmitem os dados coletados para um nó central (ex. um *smartphone*), situado na proximidade do paciente (cerca de 2 metros). O nó central, em conjunto com os nós sensores, compõe a rede corporal sem fio, sendo a comunicação entre eles do tipo intra-WBAN. Esta comunicação dentro da WBAN ocorre por meio de transmissões de baixa potência/curto alcance (ex. Zigbee, Bluetooth). Assim, neste trabalho pode-se considerar, sem perda de generalidade, que a comunicação intra-WBAN não causa interferências nas WBANs dos demais pacientes.

O nó central transmite os dados recebidos para uma estação base (comunicação inter-WBAN). A estação dá vazão a esses dados até um repositório de dados localizado em um centro médico ou na nuvem. Os dados são então armazenados e/ou analisados. É importante notar que, neste trabalho, denominamos estação base qualquer dispositivo controlador de uma rede, como exemplo, um ponto de acesso. Assim como [Doost-Mohammady and Chowdhury 2012, Phunchongharn et al. 2010], considera-se um sistema no qual a estação base não possui restrições de processamento.

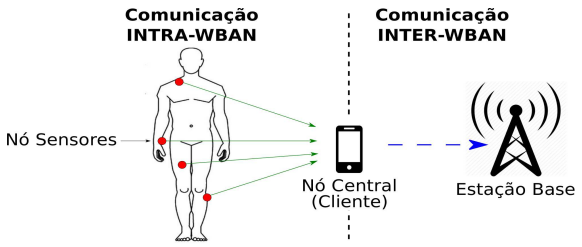


Figura 1. Visão geral do sistema.

Assume-se que a estação base tenha duas interfaces de rede sem fio: uma destinada à negociação de canais e outra à transmissão de dados. Essa característica permite a comunicação da estação base em dois canais simultaneamente, aumentando a taxa de transmissão e reduzindo o atraso na coleta dos dados [Soua and Minet 2011].

Assume-se ainda que, o nó central possui apenas uma interface de comunicação sem fio. Tanto as interfaces da estação base quanto as das WBANs podem sintonizar livremente em qualquer um dos canais disponíveis. Para uma maior economia de energia, todas as transmissões de dados ocorrem diretamente entre o nó central e a estação base, ou seja, em modo *single-hop* [Pešović et al. 2010]. Quando existirem múltiplas estações base no ambiente, assume-se que essas podem se comunicar por meio de um **canal confiável** (ex. uma conexão cabeada). Essa premissa é comum, uma vez que as estações podem se conectar por meio cabeado, com taxas de erro e latência muito inferiores às encontradas no meio sem fio. O PDAC foca na comunicação inter-WBAN, a qual é propensa a interferências. A comunicação intra-WBAN e a comunicação cabeada entre as estações base estão fora do escopo deste trabalho.

3.2. O Protocolo PDAC

No PDAC, os canais de comunicação são divididos em duas categorias: **canal de controle** e **canal de dados**. Ambos estão localizados em uma banda não licenciada do espectro de radiofrequência. As duas interfaces na estação base são denominadas de interface de controle e de dados. A interface de controle está sintonizada em um canal fixo conhecido por todos os nós centrais, enquanto a interface de dados sintoniza de forma adaptativa dentre os canais de dados disponíveis. Durante a comunicação inter-WBAN, a interface de controle é utilizada para negociar um canal de dados.

Um dos diferenciais do PDAC consiste na escolha apropriada e na alocação de canais de maneira distribuída e coordenada entre as estações base. A alocação de canais foi mapeada para um problema de coloração de grafos, onde dois vértices adjacentes não podem ser coloridos com a mesma cor. Suponha um grafo $G = (V, E)$, os vértices no conjunto V representam as estações base, e as arestas no conjunto E representam conflitos entre os canais utilizados pelas estações base. Cada cor representa um canal. Com base em uma solução gulosa, o PDAC evita de maneira distribuída as interferências e proporciona uma divisão de carga entre as estações base.

Cada estação base mantém um controle de quais canais estão ocupados e livres com base em um histórico de alocação. Dessa maneira, a estação tem pleno controle sob o atual estado do canal. Seguindo o PDAC, o acesso ao meio é composto por duas fases. A primeira é chamada de **fase de controle**. Nela, o nó central negocia com a estação base

um canal de dados. Ao receber a requisição do nó central, a estação base envia um pacote para as outras através do canal confiável existente entre as estações informando que possui a intenção de alocar um determinado canal de dados. Caso não haja impedimentos (ex. todos os canais de dados ocupados), a estação base aloca o canal, enviando um pacote para a WBAN requisitante com tal informação, encerrando assim a fase de controle.

A **fase de transmissão** ocorre após a alocação do canal. O nó central sintoniza no canal negociado e espera um pacote para iniciar a transmissão dos dados. Nesse momento, a interface de dados sintoniza no canal alocado e envia o pacote esperado pelo nó central. Recebidos os dados, a estação base envia o pacote de confirmação para o nó central e encerra a comunicação. A estação base anuncia para as outras estações que o canal de comunicação está livre e pode ser alocado para outras comunicações (detalhes são apresentados nas próximas subseções).

De modo geral, o protocolo traz uma abordagem de alocação híbrida de canais. Ele mantém uma interface da estação base em um canal estático e aloca um canal para a outra interface através de uma estratégia dinâmica. Essa abordagem tem o objetivo de iniciar uma comunicação entre a estação base e o nó central para evitar problemas de *surdez*, comuns em uma abordagem totalmente dinâmica. A partir desse contato inicial, o transmissor e receptor negociam qual canal de dados utilizar, evitando maiores atrasos quando comparado a um processo de tentativa e erro para encontrar qual canal a estação base está sintonizada [Lee et al. 2011]. Esta abordagem não necessita qualquer sincronização de tempo entre a estação base e o nó central.

Entretanto, a alocação híbrida de canais resulta em duas desvantagens. Primeiro, o canal de controle pode vir a tornar um gargalo quando o número de canais de dados disponíveis é muito grande. Segundo, um canal poderá ser subutilizado, uma vez que o canal de controle é utilizado apenas para controle. Como forma de contornar o problema de baixa utilização dos canais, um mecanismo de agregação de canais é proposto. De maneira geral, utilizando a agregação de canais, a estação base passa a alocar uma determinada quantidade de canais adjacentes, se disponível, para o nó central. Dessa forma, utilizamos uma técnica de agregação de canais para melhorar a largura de banda, minimizar atrasos e aumentar, de maneira geral, as taxas de transmissão [Rehmani et al. 2012].

3.2.1. Fases de controle e de transmissão

A Figura 2 apresenta a sequência de mensagens do protocolo PDAC. No PDAC, cada estação mantém três estruturas de dados. A primeira é a lista de canais disponíveis (LCD), contendo todos os canais disponíveis para uso pela estação base. A segunda estrutura é a lista de canais utilizados (LCU), que representa todos os canais já alocados naquela estação. E por fim, a estação base mantém uma lista de canais bloqueados (LCB) cujo objetivo é manter o controle de quais canais estão usados por outras estações base.

A fase de controle é caracterizada pelo contato inicial entre um nó central desejando transmitir os dados e uma estação base (mensagens 1, 2, 3 e 4 em vermelho na Figura 2). Esta fase é iniciada quando o nó central envia um pacote de requisição RTS (mensagem 1) para a estação base. O PDAC não considera nenhum tipo de diferenciação de serviço, logo todos os nós centrais acessam ao canal de controle com direitos iguais seguindo o protocolo de acesso ao meio CSMA/CA. A interface de controle da estação recebe esta requisição e busca em sua lista LCD um canal disponível. Caso a estação base

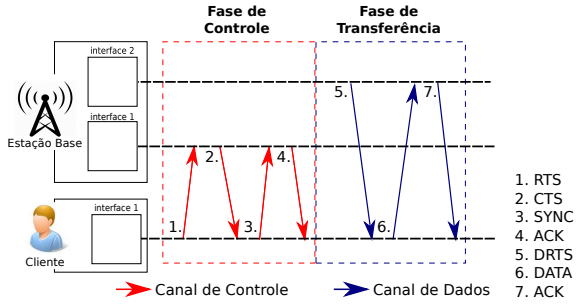


Figura 2. Diagrama de sequência do protocolo PDAC.

não encontre um canal disponível, ou seja, se a lista LCD estiver vazia, a estação envia um pacote de negação chamado de NCTS indicando impossibilidade de alocar um canal para a transmissão. Ao receber esse pacote de negação, o nó central espera um tempo aleatório para realizar uma nova requisição. Caso a estação base encontre um canal, esse canal é retirado da LCD e adicionado na LCU, indicando que o canal está sendo usado. A estação base então anuncia o canal alocado para as outras estações base, iniciando a fase de sincronização entre as estações (detalhes na Subseção 3.2.2).

Após o canal ser confirmado na fase de sincronização, o canal alocado (ou grupo de canais, detalhes na Seção 3.2.3) é enviado para o nó central via CTS. Após receber o CTS, o nó central envia um pacote SYNC (mensagem 3) confirmando o recebimento do canal. A estação base, ao receber esse pacote, envia um pacote de ACK (mensagem 4) confirmando que em breve irá se comunicar com o nó central no canal alocado. Ao receber o ACK, o nó central sintoniza no novo canal, encerrando assim a fase de controle.

É importante observar que essa fase é bastante similar a mecanismos clássicos de RTS/CTS já utilizados na literatura, onde é necessária a adição de um campo no cabeçalho do pacote CTS usado para o envio do canal alocado e outro campo representando a quantidade de canais adjacentes que foram alocados. O tamanho desses cabeçalhos varia de acordo com a quantidade de canais disponíveis para a alocação e quantos canais adjacentes podem ser alocados.

A fase de transmissão é iniciada quando um canal é adicionado na LCU e segue as mensagens 5, 6, 7 e 8 em azul na Figura 2. Todos os canais presentes na LCU possuem um nó central esperando por um pacote DRTS para iniciar a transmissão. A segunda interface utiliza a LCU como uma fila de sintonização. A cada canal sintonizado, a estação base envia um pacote DRTS (mensagem 5) anunciando que está pronta para a recepção dos dados. Após receber este pacote, o nó central inicia a transmissão dos dados (mensagem 6). Após receber os dados, a estação base envia um pacote de confirmação ACK (mensagem 7), retira o canal utilizado da LCU e inclui na LCD. A estação base então envia um pacote com o canal liberado para as outras estações base iniciando, novamente, a fase de sincronização entre estações (Subseção 3.2.2). Se ao fim desse processo a LCU estiver vazia, a interface de dados entra em um modo ocioso. Caso a LCU ainda possua elementos, a fase de transmissão é reiniciada.

3.2.2. Sincronização entre estações

A sincronização entre as estações ocorre após a estação base receber um RTS e após uma transmissão de dados bem sucedida, isto é, após um ACK ser enviado para o nó central. Como mencionado anteriormente, o PDAC segue uma solução para o problema

tradicional de coloração de grafos. No entanto, algumas alterações do problema clássico foram realizadas. A primeira alteração foi a adição de uma cor neutra. Consideramos a cor neutra sendo a única cor permitida a ser usada em dois nós adjacentes. No PDAC, um vértice com a cor neutra representa uma estação base com a interface de dados ociosa. Outra mudança é a possibilidade de um vértice estar colorido com mais de uma cor. Com essa alteração, assumimos a possibilidade da estação base alocar mais de um canal para a transmissão dos dados.

Ao inicio do protocolo, toda estação base está marcada somente com a cor neutra. Cores neutras adjacentes refletem contato entre estações base ociosas, que por definição, não apresentam conflitos. O protocolo apresentado, ao alocar um canal, substitui a cor neutra de uma estação por outra cor. Para realizar tal coloração, utililzamos um algoritmo guloso de coloração de grafos, composto de duas operações: a adição de cor a um vértice e remoção de uma cor. Quanto uma estação base perde todas as suas cores, atribuimos a ela novamente a cor neutra.

O protocolo PDAC segue com duas operações básicas nas estações base: a alocação de canais e a liberação de canais. A alocação de canais ocorre no momento em que a estação base recebe um RTS. Após receber uma requisição, a estação base busca um canal (ou um conjunto de canais, detalhes na Subseção 3.2.3) em sua LCD. Caso a busca retorne sucesso, a estação base envia uma mensagem de bloqueio BLOCK contendo os canais bloqueados para as estações base adjacentes.

A estação base receptora desse pacote de bloqueio adiciona esses canais na lista LCB. Esses canais serão retirados somente após receber um pacote de liberação da estação base de origem do pacote de bloqueio. Todo canal presente na LCB não pode ser utilizado na LCD. Uma estação base pode ter os mesmos canais bloqueados por mais de uma estação base. Neste caso, esse canal poderá ser utilizado somente após a liberação por todas as estações base. Após uma transmissão bem sucedida, a estação base envia um pacote de liberação FREE com os canais utilizados na transmissão. A estação base, após receber esse pacote, retira todos os canais bloqueados recebidos dessa estação base. Caso não haja nenhuma outra estação bloqueando esse canal, ele poderá ser utilizado na LCB.

O protocolo PDAC não garante a solução ótima. A estratégia gulosa previne que estações base adjacentes utilizem o mesmo canal de forma simultânea. Assim, com a comunicação entre as estações base, pode-se dizer que o PDAC evita que nós centrais causem interferência entre si na transmissão dos dados. Além disso, como o número de frequências disponíveis é relativamente baixo, um custo computacional baixo é esperado. Desta forma, pode-se dizer que o PDAC é capaz de prover uma transmissão livre de interferências, independente do número de estações base.

3.2.3. Agregação de Canais no PDAC

Neste trabalho, a estação base possui a função de gerenciar o estado atual do espectro de radiofrequência, além de servir como destino local para os dados das WBANs. No protocolo PDAC, durante o gestão do espectro, a estação base divide o estado dos canais em três categorias: (1) canais livres, (2) canais ocupados e (3) canais bloqueados. Os canais livres representam canais que podem ser alocados para a transmissão de dados, os canais ocupados representam canais que já estão alocados naquela estação base e os canais bloqueados são canais sendo utilizados por outras estações base. Divide-se a categoria de

canais livres em duas subcategorias: (1-1) canais de borda e (1-2) canais centrais. Os canais de borda são canais adjacentes de canais bloqueados ou ocupados (suscetíveis a interferências de sobreposição de canais), e os canais centrais são adjacentes de canais de borda ou de outros canais centrais. Ao receber uma requisição de canal pelo nó central, o objetivo da estação base é oferecer a maior banda possível, evitando ao máximo a alocação de canais de borda (evitando interferências de canais sobrepostos). O canal de borda mantém uma distância segura entre dois blocos de canais alocados.

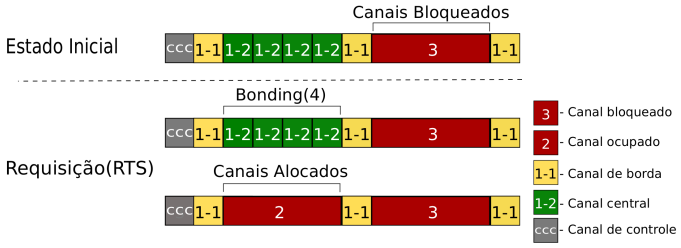


Figura 3. Exemplo do processo de gerenciamento, união e alocação de canais após uma requisição (RTS) de um nó central, sendo que no máximo 4 canais podem ser unidos por vez.

Consideramos elegíveis para as técnicas de agregação de canais apenas os canais centrais. A Figura 3 ilustra o processo de alocação e de bloqueio de canais. Após receber uma requisição do nó central, a estação base busca por k canais centrais disponíveis. Caso encontre, a estação aplica técnicas de agregação de canais e aloca esse canal agrupado para o nó central. Caso não encontre, a estação base realiza uma nova busca decrementando o k, sendo assim, desta vez buscando $k - 1$ canais centrais disponíveis. O processo se repete até que a estação base encontre uma parcela de canais disponíveis ou quando k for igual a 0, significando que não existe nenhum canal central disponível. Neste momento, a estação base inicia uma busca por um canal de borda para a alocação. Se encontrado, esse canal é alocado para a comunicação. Caso contrário, a estação base nega a comunicação com o nó central alegando que não existem canais disponíveis para a comunicação.

4. Avaliação do Sistema

O protocolo PDAC foi avaliado através de simulações de cenários de um ambiente hospitalar utilizando o Castalia [Boulis et al. 2011]. O Castalia é um simulador de redes baseado no OMNeT++ [Varga et al. 2001] e é utilizado por pesquisadores para testes de algoritmos e protocolos com base em modelos reais de canal sem fio e rádio. O protocolo PDAC, neste trabalho, foi implementado sobre o protocolo T-MAC [Van Dam and Langendoen 2003], porém o PDAC pode ser implementado sob qualquer protocolo que se utilize de RTS/CTS.

Neste trabalho, consideramos que cada nó central de uma WBAN se comunica com a estação base utilizando o rádio Zigbee CC2420 operando na faixa de 2,4 Ghz. Esse rádio é capaz de operar em até 16 canais diferentes com 20 MHz de largura, cada. Nas simulações, foi fixada a largura de banda de cada canal para até 250 kbps. Note que, durante as simulações, variamos o número de canais disponíveis (4, 8 e 16 canais disponíveis) para avaliar o desempenho do novo protocolo em função dos canais disponíveis. Variamos também o número de canais que podem ser agregados (2, 4 e 8 canais).

Devido a limitações no simulador, interferências entre os canais foram desconsideradas. Consideramos cada campo adicionado no pacote CTS com o tamanho de 4 bits, cada.

As simulações são conduzidas em dois ambientes médicos realistas: um departamento de emergência sobrecarregado e um ambiente calmo, como uma sala de radiologia. Os dois cenários contam com uma área de 30 m^2 e estações base fixas de forma que não ocorra problemas de terminal oculto. Os pacientes e seus nós centrais são distribuídos no ambiente seguindo uma distribuição uniforme e estão no raio de alcance das estações base. A Tabela 1 sumariza as principais características de cada aplicação médica. Como esperado, cada cenário possui suas próprias características e maiores detalhes podem ser encontrados em [Baker and Hoglund 2008].

Tabela 1. Aplicações médicas e carga de dados.

Aplicação médica	Nós centrais Dpt. Emergência	Nós Centrais Radiologia	Pacotes /s	Tamanho Pacote (kb)
Diagnóstico	3	1	5	5,1
Telemetria	12	9	5	2,6
Bomba de Infusão	10	10	1	1,0

Durante as simulações, os dispositivos de todos os pacientes possuem a mesma prioridade no acesso ao canal. Cada dispositivo gera um fluxo de dados de acordo com sua aplicação (vide Tab. 1). Os nós centrais são iniciados de maneira sequencial de tal forma que a cada 2 segundos um novo nó central é adicionado ao ambiente simulado. Ao término da simulação, todos os nós centrais são encerrados ao mesmo tempo.

Cada simulação dura 16 minutos e o minuto inicial é descartado em função do período transiente da simulação. A não ser que mencionado de outra forma, os resultados apresentados são referentes a 30 execuções de cada ambiente simulado, com um nível de confiança de 95%. Para analisar os desempenhos dos protocolos simulados, as seguintes métricas foram utilizadas:

• **Latência:** o tempo médio gasto para um pacote ser enviado de uma WBAN para a estação base, incluindo tempos de propagação, transmissão, processamento e em filas.

• ***Goodput*:** a proporção média entre a quantidade de dados entregues e o tempo decorrido para sua entrega. Consideram-se neste sentido apenas dados úteis de fato, sendo desconsideradas as retransmissões e os cabeçalhos dos protocolos.

• **Taxa de perda de pacotes:** a proporção média de pacotes que foram recebidos incorretamente ou descartados por *timeout* entre todos os pacotes enviados. De acordo com [Baker and Hoglund 2008], considera-se que aplicações médicas necessitam de taxa de perda de pacotes inferior a 0,01 e latência inferior a 200 ms.

4.1. Resultados

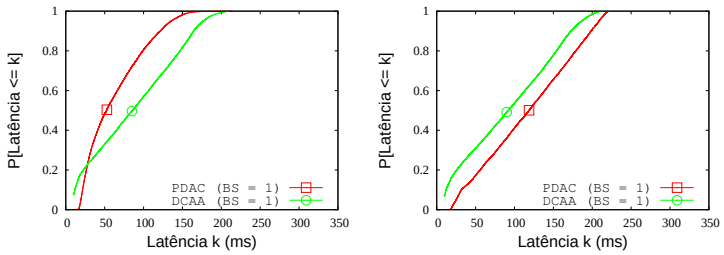
Inicialmente, comparamos o protocolo PDAC com o DCAA [Lee et al. 2011], um dos protocolos mais representativos da literatura quando se refere à alocação dinâmica de canais. Segundo, nós avaliamos o impacto do número de estações base disponíveis no desempenho do novo protocolo. Por último, avaliamos a melhora de desempenho do PDAC, quando utilizada a agregação de canais.

PDAC versus DCAA

Avaliamos o PDAC e o comparamos ao DCAA nos dois cenários médicos descritos anteriormente. De forma a realizar uma comparação justa, limitamos o PDAC a uma configuração equivalente à suportada pelo DCAA. Neste caso, o PDAC não faz uso de múltiplas estações e ambos os protocolos estão limitados ao uso de 4 canais de dados. Alteramos o número de canais disponíveis e as conclusões apresentadas se mantém. Omitimos tais resultados por restrições de espaço.

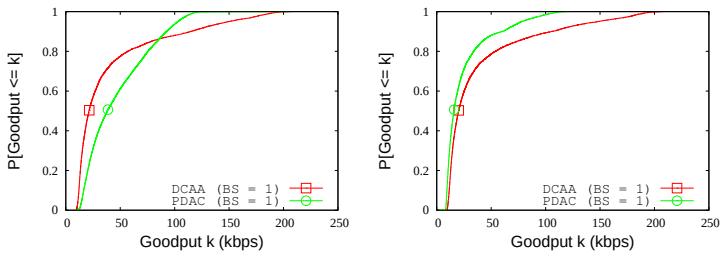
A Figura 5 apresenta a função de distribuição de probabilidade acumulada da latência em dois cenários distintos, um departamento de emergência e uma sala de radiologia. Observe que, para um cenário de carga de dados baixa (sala de radiologia), o PDAC apresenta desempenho melhor que o DCAA. Neste caso, o novo protocolo tem uma latência média 1,4 vezes melhor que o DCAA. O PDAC apresentou uma taxa de entrega de 99,9% dos pacotes, enquanto o DCAA apresentou menos de 88% de taxa de entrega. Mais ainda, em mais de 45%, o *goodput* apresentado pelo PDAC foi de pelo menos 50 kbps, enquanto que em menos de 25% dos casos o DCAA atinge esta marca.

Em um cenário de carga de dados alta, como em um departamento de emergência médica, há uma degradação do desempenho em ambos os protocolos. Mais ainda, os pacotes entregues pelo DCAA têm latência média menor que a do novo protocolo. Porém, a taxa de perda que o PDAC apresenta é menor que o DCAA. Enquanto este apresentou cerca de 30% de perda, o PDAC teve em torno de 24%. Os valores de *goodput* encontrados seguem a mesma tendência da latência. Cerca de 25% dos casos no DCAA apresentam mais de 50 kbps, enquanto cerca de 18% do PDAC atingem a mesma marca.



(a) Radiologia (b) Dpt. Emergência

Figura 4. Distribuição de latência apresentada pelos protocolos DCAA e PDAC.



(a) Radiologia (b) Dpt. Emergência

Figura 5. Distribuição de *goodput* apresentada pelos protocolos DCAA e PDAC.

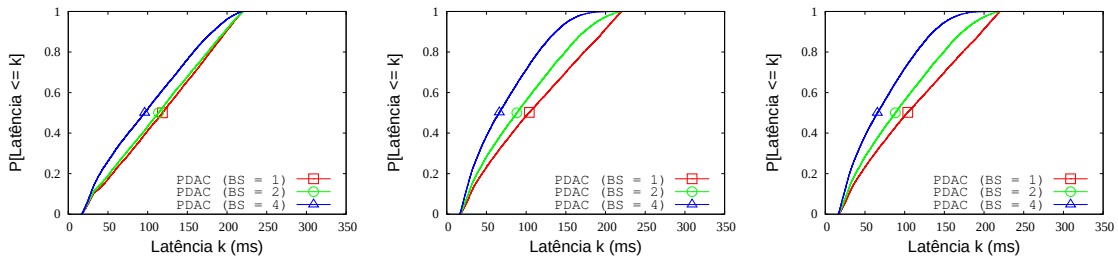
Em suma, apesar da melhoria da latência em um ambiente com baixa carga, os protocolos avaliados não conseguem atender às demandas de aplicações médicas em um

cenário com alta demanda de recursos. Neste sentido, há evidências que o uso de uma única estação base pode não ser apropriado em um ambiente médico-hospitalar.

Impacto de múltiplas estações e múltiplos canais

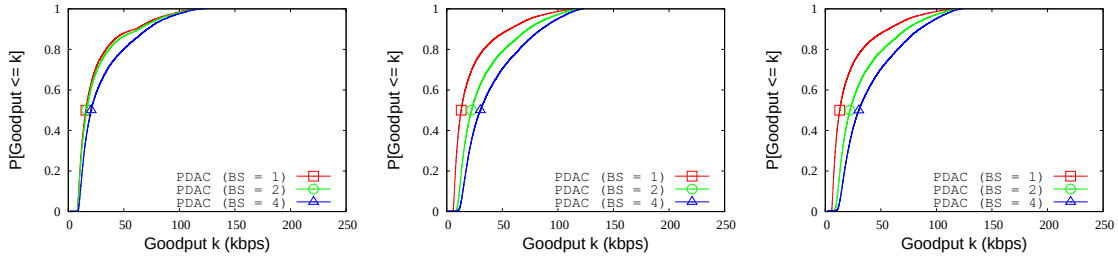
A principal característica do PDAC é o uso de múltiplas estações base, distribuindo a carga entre elas. Assim, nós avaliamos seu comportamento variando o número de estações base até 4, onde o grafo de interferências entre as estações base é completo, e o número de canais disponíveis para comunicação até 16. Nesta análise, o cenário de radiologia foi desconsiderado uma vez que uma única estação base foi capaz de atender seus requisitos. As Figuras 6 e 7 apresentam a função de distribuição de probabilidade acumulada da latência e do *goodput*, respectivamente. De forma geral, quanto maior o número de estações base utilizada, melhor é o desempenho do novo protocolo. Da mesma forma, o aumento no número de canais disponíveis para comunicação gera ganhos consideráveis. Por exemplo, a partir de duas estações base, com 8 canais disponíveis para transmissão de dados, o PDAC foi capaz de atender aos requisitos das aplicações médicas, apresentando uma taxa de entrega de pacotes de 99.9%. Neste caso, a latência média foi de 86,45 ms e o *goodput* foi de 33,36 kbps.

Apesar do ganho que ambos os fatores trazem, notamos que o crescimento do número de canais tem menor impacto no desempenho que o número de estações. Ao se aumentar o número de estações, esperava-se que ocorresse uma divisão de carga entre elas. Por outro lado, mesmo com a existência de múltiplos canais, a fase inicial do protocolo não é paralelizável, o que limita os ganhos dentro de uma mesma estação.



(a) 4 Canais (b) 8 Canais (c) 16 Canais

Figura 6. Latência em relação ao número de canais disponíveis.



(a) 4 Canais (b) 8 Canais (c) 16 Canais

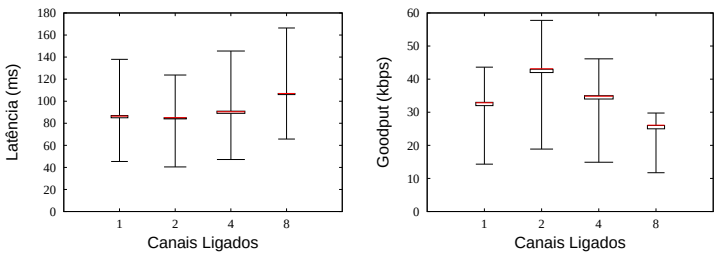
Figura 7. *Goodput* em relação ao número de canais disponíveis.

Impacto do uso da agregação de canais

Técnicas de agregação de canais podem melhorar a comunicação, quando utilizadas em um cenário de maior disponibilidade de canais de dados [Bukhari et al. 2016]. Para demonstrar o poder desta técnica, nós fixamos em 16 o número de canais de dados disponíveis e variamos o número máximo de canais que podem ser unidos na tentativa de melhorar o desempenho do protocolo. Outras configurações apresentam resultados qualitativamente semelhantes. Por restrições de espaço, omitimos estas outras análises.

As Figuras 8 (a-b) apresentam o intervalo de confiança, primeiro e terceiro quartis dos valores de latência e *goodput*, respectivamente. Tanto o comportamento da latência, quanto do *goodput* apresentam melhoras até um determinado número de canais que podem ser unidos, com posterior degradação. De fato, com um grande número de canais sendo unidos, pode ocorrer um fenômeno conhecido como inanição. O nó central que conseguir alocar e unir muitos canais terá seu desempenho melhorado, mas os demais nós centrais não conseguirão um conjunto de canais livres para suas transmissões. Estes nós centrais contribuirão de forma negativa para o desempenho geral do sistema.

O ganho na latência não é tão expressivo quanto o ganho de *goodput*. De fato, o ganho na latência é apenas marginal uma vez que um determinado pacote continua sendo transmitido por um dos canais disponíveis do conjunto utilizado na ligação. Por outro lado, vários pacotes são transmitidos em paralelo e, mesmo com uma porção serial no protocolo, o ganho pode ser superior a 33%.



(a) Latência (b) *Goodput*

Figura 8. Avaliação do atraso e *goodput* em relação ao máximo de canais unidos.

5. Conclusões e Trabalhos Futuros

Este trabalho apresentou o PDAC, um protocolo da camada MAC ciente dos requisitos das aplicações médicas para alocar dinamicamente canais em cenários hospitalares densos. O PDAC oferece uma solução para reduzir interferências. Ele permite a cooperação de múltiplas estações base no mesmo ambiente médico-hospitalar. O protocolo foi avaliado através de simulações de dois ambientes médicos realistas: uma sala de radiologia e um departamento de emergência. Sob a presença de uma única estação base, o PDAC é capaz de melhorar o desempenho da comunicação entre a estação base e os dispositivos em cenários com menor carga. Em um cenário de alta carga, as perdas de pacote são desprezíveis sob a utilização de múltiplas estações base. De fato, os resultados das simulações indicam que, em um ambiente médico realista, o PDAC é capaz de, em média, aumentar a vazão em 30% e reduzir a latência em 40%, quando comparado com outros protocolos de alocação de canais representativos existentes na literatura.

Como trabalhos futuros, espera-se desenvolver mecanismos cientes dos níveis de energia nos dispositivos médicos com o intuito de estender a vida útil destes dispositivos. Mais ainda, esperamos oferecer mecanismos de diferenciação de serviço a partir do nível de criticidade de eventos da aplicação médica e verificar a convivência do protocolo proposto com outras tecnologias operando nos mesmos canais.

Referências

Baker, S. D. and Hoglund, D. H. (2008). Medical-grade, mission-critical wireless networks [designing an enterprise mobility solution in the healthcare environment. *IEEE Eng. in Medicine and Biology Magazine*, 27(2):86–95.

Boulis, A. et al. (2011). Castalia: A simulator for wireless sensor networks and body area networks. *NICTA: National ICT Australia*.

Bukhari, S. H. R., Rehmani, M. H., and Siraj, S. (2016). A survey of channel bonding for wireless networks and guidelines of channel bonding for futuristic cognitive radio sensor networks. *IEEE Commun. Surveys & Tutorials*, 18(2):924–948.

Doost-Mohammady, R. and Chowdhury, K. R. (2012). Transforming healthcare and medical telemetry through cognitive radio networks. *IEEE Wireless Commun.*, 19(4):67–73.

Fang, G., Dutkiewicz, E., Yu, K., Vesilo, R., and Yu, Y. (2010). Distributed inter-network interference coordination for wireless body area networks. In *IEEE GLOBECOM*, pages 1–5.

Kyasanur, P. and Vaidya, N. H. (2006). Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks. *ACM SIGMOBILE Review*, 10(1):31–43.

Lee, B., Yun, J., and Han, K. (2011). Dynamic channel adjustable asynchronous cognitive radio mac protocol for wireless medical body area sensor networks. In *Communication and Networking*, volume 266, pages 338–345. Springer.

Movassaghi, S., Abolhasan, M., Lipman, J., Smith, D., and Jamalipour, A. (2014). Wireless body area networks: A survey. *IEEE Commun.Surveys & Tutorials*, 16(3):1658–1686.

Omeni, O., Wong, A. C. W., Burdett, A. J., and Toumazou, C. (2008). Energy efficient medium access protocol for wireless medical body area sensor networks. *IEEE Trans. on biomedical circuits and systems*, 2(4):251–259.

Pešović, U. M., Mohorko, J. J., Benkič, K., and Čučej, Ž. F. (2010). Single-hop vs. multi-hop–energy efficiency analysis in wireless sensor networks. In *Telecommunications Forum*.

Phunchongharn, P., Hossain, E., Niyato, D., and Camorlinga, S. (2010). A cognitive radio system for e-health applications in a hospital environment. *IEEE Wireless Commun.*, 17(1):20–28.

Rehmani, M. H., Lohier, S., and Rachedi, A. (2012). Channel bonding in cognitive radio wireless sensor networks. In *International Conference on Selected Topics in Mobile and Wireless Networking (iCOST)*, pages 72–76. IEEE.

Soua, R. and Minet, P. (2011). A survey on multichannel assignment protocols in wireless sensor networks. In *IFIP Wireless Days (WD)*, pages 1–3. IEEE.

Van Dam, T. and Langendoen, K. (2003). An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 171–180. ACM.

Varga, A. et al. (2001). The omnet++ discrete event simulation system. In *Proceedings of the European simulation multiconference (ESM'2001)*, volume 9, page 65. sn.

Yu, J.-Y., Liao, W.-C., and Lee, C.-Y. (2006). A MT-CDMA based wireless body area network for ubiquitous healthcare monitoring. In *IEEE Biomedical Circuits and Systems Conference*, pages 98–101. IEEE.

Maximizando a Vazão Através de Múltiplos Caminhos em Plataformas com Dois Rádios

Nildo dos Santos Ribeiro Júnior,
Marcos A. M. Vieira, Luiz F. M. Vieira

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais – Belo Horizonte, MG – Brasil

{nildo,mmvieira,lfvieira}@dcc.ufmg.br

Abstract. *Applications for Wireless Sensor Networks are requiring higher throughput. Platforms with two radios were proposed to achieve it, conserving the energy efficiency. There is a bulk-data transfer protocol optimized for dual-radio platforms proposed in the literature, but it doesn't use all the hardware resources available. In this work we propose a new multi-hop bulk-data transfer protocol for wireless networks that simultaneously uses two disjoint paths, using all available radios. The protocol was implemented using TinyOS and evaluated in the physical world. We compare our proposal with FastForward, the state-of-the-art protocol for dual-radio. Our approach doubled the throughput getting very close to the maximum theoretical limit value.*

Resumo. *Aplicações para Redes de Sensores Sem Fio estão requerendo maior vazão. Foram desenvolvidas plataformas com dois rádios para aumentar a vazão conservando a eficiência energética. Há uma proposta de protocolo de transferência de dados em massa otimizado para essas plataformas, porém ele não aproveita todo o recurso de hardware disponível. Nesse trabalho propomos um protocolo de transferência massiva de dados em vários saltos para redes sem fio que utiliza dois caminhos distintos simultaneamente, aproveitando os dois rádios disponíveis nos nós. O protocolo foi implementado no TinyOS e avaliado experimentalmente no mundo físico. Comparamos nossa proposta com o FastForward, o estado da arte. Nossa abordagem dobrou a vazão chegando próximo ao limite máximo teórico.*

1. Introdução

As Redes de Sensores Sem Fio (RSSFs) consistem em conjuntos de nós sensores que agem tanto como coletores de dados quanto como retransmissores de pacotes na rede. Cada nó contém um ou mais sensores, um microprocessador e um ou mais transceptores de rádio. De acordo com [Akyildiz and Vuran 2010], a comunicação sem fio provê a vantagem de facilidade de implantação, e a capacidade de sensoreamento distribuído faz com que RSSFs tenham potencial para serem um importante componente na nossa vida diária. Elas têm muitas aplicações, como em monitoração ambiental, na agricultura, em assistência médica e em construções inteligentes.

O projeto de uma RSSF depende significativamente da aplicação. O ambiente, os princípios de design da aplicação, o hardware e as restrições de projeto podem ser muito diferentes em cada cenário [Yick et al. 2008]. Por exemplo, o ambiente é importante para

determinar o tamanho, o esquema de implantação e até mesmo a topologia da rede. Porém quando estamos falando sobre o projeto de hardware dos nós sensores, há dois problemas chave levados em consideração: o custo e o consumo de energia.

Como as RSSFs geralmente consistem em uma grande quantidade de nós sensores, minimizar o custo de cada unidade é muito importante para que o custo total da rede não seja muito elevado. Essa restrição faz com que a maioria das plataformas de RSSFs tenham muito pouco poder de processamento e memória. Como um nó sensor geralmente é alimentado por pilhas ou baterias, o consumo de energia deve ser minimizado para prolongar a vida útil deles. Para economizar energia, deve-se minimizar a quantidade de transmissões sem fio, já que o rádio consome uma quantidade muito grande de energia comparada ao resto dos componentes da arquitetura de hardware.

Minimizar o custo e o consumo de energia dos nós sensores faz com que o desempenho da rede seja reduzido. Esse compromisso entre custo e desempenho é muito importante no design de RSSFs. De acordo com [Jurdak et al. 2011], o design de plataformas tradicionais de RSSFs favoreceram o baixo consumo de energia ao custo da vazão ou alcance da rede. Isso fez sentido no contexto em que se deu o início do desenvolvimento de RSSFs, onde a maioria das aplicações era de coletar pequenas quantidades de dados, como valores de temperatura ou iluminação.

Hoje em dia, aplicações também estão sendo desenvolvidas para coletar dados de som e vídeo, que têm uma demanda por alta vazão na rede. Então embora o baixo consumo de energia ainda seja importante no design de RSSFs, a vazão tem ganhado importância nesses novos tipos de aplicações. Levando ainda em consideração que novas tecnologias de captação de energia têm sido desenvolvidas, o projeto das plataformas pôde passar a priorizar um pouco mais outros fatores, além do consumo de energia. Uma boa opção é priorizar a eficiência energética na transmissão, que é a quantidade de energia utilizada para transmitir uma certa quantidade de dados. Dessa maneira, nós podemos ter aplicações com maior vazão que poderão consumir mais energia no total, porém conservando a eficiência energética do sistema, pois são capazes de transmitir uma quantidade muito maior de dados.

Essa mudança de paradigma motivou pesquisadores a desenvolver novas plataformas para RSSFs, com o objetivo de prover maior vazão continuando energeticamente eficiente. Uma técnica que permite alcançar esse objetivo é prover diversidade de rádio, que significa ter mais de um rádio em cada nó sensor. Se cada rádio opera em uma banda diferente, eles não vão interferir um com o outro enquanto estão comunicando, reduzindo a perda de pacotes devido à interferência e permitindo transmissões simultâneas entre nós sensores. De acordo com [Kusy et al. 2011], a diversidade de rádio pode melhorar a taxa de entrega, a estabilidade da rede e os custos de transmissão por um pequeno aumento no custo energético de um único rádio.

Nesse artigo, apresentamos um protocolo de transferência massiva de dados em vários saltos para RSSFs, projetado para plataformas que possuem dois rádios operando em frequências diferentes. O protocolo utiliza dois caminhos distintos para fazer a comunicação entre dois nós na rede. Os pacotes são divididos entre esses dois caminhos e transmitidos simultaneamente, alternando o rádio utilizado em cada salto do caminho e aproveitando os dois rádios disponíveis em todos os nós sensores. Na prática, ele con-

segue aumentar a vazão na rede em até 67%, quando comparado com um protocolo que utiliza apenas um caminho.

O restante do artigo está organizado da seguinte maneira: na seção 2 são apresentados alguns conceitos básicos e os trabalhos relacionados. Na seção 3 é apresentado o protocolo proposto, e na seção 4 é descrito o problema dos caminhos disjuntos com paridade. A seção 5 apresenta a modelagem do problema usando programação linear inteira. Na seção 6 são explicados os detalhes de implementação de teste do protocolo. Na seção 7 são descritos os experimentos realizados para avaliar o desempenho do protocolo e discutidos os resultados. Finalmente, as conclusões e trabalhos futuros são apresentados na seção 8.

2. Trabalhos Relacionados

Os protocolos tradicionais de coleta de dados em RSSFs não suportam alta vazão, pois eles eram projetados para minimizar o consumo de energia dos nós sensores. No cenário tradicional de plataformas com um único rádio, foram desenvolvidos vários protocolos específicos para transferência massiva de dados, por exemplo o Flush [Kim et al. 2007], FlushMF [Tavares et al. 2016] e o PIP [Raman et al. 2010]. Esses protocolos estabelecem uma rota de um único caminho entre um nó fonte e um nó de destino, desabilitam o ciclo de trabalho do rádio e forçam um escalonamento de pacotes ótimo de ponta-a-ponta nesse caminho. Até então o entendimento era que o baixo consumo de energia deveria ser deixado de lado para atingir alta vazão na rede.

Um pouco mais tarde é apresentado o Burst Forward [Duquennoy et al. 2011], uma técnica que permite que o rádio dos nós intermediários no caminho mantenham os seus ciclos de trabalho ativos para economizar energia, sendo necessário que apenas o nó fonte e o nó de destino o desabilitem. Os autores mostram que mesmo assim conseguem atingir uma vazão comparável aos protocolos anteriores, utilizando uma mistura da técnica de transmissão em rajadas e da técnica de armazenamento e encaminhamento. Porém, por terem sido desenvolvidos para plataformas de um único rádio, todos esses protocolos sofrem do problema fundamental em ter apenas um rádio: não é possível transmitir e receber pacotes ao mesmo tempo. Com isso, a vazão total na rede fica limitada a até no máximo 50% da capacidade do canal de comunicação.

Com o desenvolvimento de plataformas com dois rádios, passa a ser possível transmitir e receber pacotes ao mesmo tempo. Visando aproveitar essa nova possibilidade foi desenvolvido o FastForward [Ekbatanifard et al. 2013], até então o único protocolo de transferência massiva de dados projetado para plataformas com mais de um rádio. No FastForward, os pacotes são transmitidos de um nó fonte para um nó de destino através de um único caminho entre eles. Porém, a utilização dos rádios ao longo dos saltos desse caminho é alternada, sendo que cada nó intermediário recebe pacotes por um rádio e, simultaneamente, transmite pacotes pelo outro rádio. Com isso, o limite teórico para a vazão utilizando esse protocolo passa a ser 100% da capacidade do canal.

No FastForward, o fato de utilizar dois rádios em bandas diferentes já ajuda a resolver o problema de interferência entre pacotes. Além disso, ele utiliza a técnica já utilizada pelos protocolos anteriores de alternar canais entre os rádios de uma mesma banda, o que ajuda a diminuir ainda mais o efeito da interferência entre as transmissões. A Figura 1 mostra o esquema de alocamento de rádios e canais utilizado no FastForward.

Nele, duas transmissões sendo feitas pelo mesmo rádio e pelo mesmo canal estarão a pelo menos três saltos de distância um do outro.

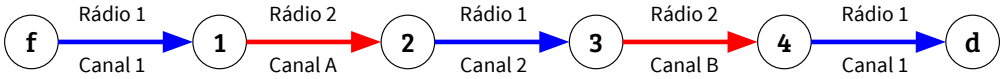


Figura 1. Esquema de alocamento de rádios e canais utilizado no FastForward

Observe que, no FastForward, os nós intermediários utilizam todo o recurso de rádio disponível, pois estão sempre recebendo pacotes por um rádio e transmitindo pelo outro. Porém, os nós fonte e de destino utilizam apenas metade do recurso de rádio disponível. O nó fonte transmite apenas por um rádio, e o nó de destino recebe por apenas um rádio. Para passar a aproveitar os dois rádios também nesses nós, é que foi desenvolvida a abordagem apresentada na próxima seção, que utiliza dois caminhos para a transmissão de dados.

3. Protocolo

Nessa seção, é apresentada a descrição de um novo protocolo de transmissão massiva de dados em vários saltos para RSSFs que utilizam plataformas com dois rádios, e utiliza dois caminhos distintos na rede para transmitir pacotes simultaneamente. O protocolo foi projetado para atuar juntamente com os outros protocolos tradicionais de RSSFs, que utilizam de um modo de economia de energia. Quando a aplicação precisar fazer uma transferência massiva de dados, ela chama o protocolo, que configura os caminhos a serem utilizados e faz com que esses nós saiam do modo de economia de energia, e fiquem no modo de transmissão até que a transferência seja concluída com alta vazão.

O princípio básico do protocolo proposto é utilizar 100% dos recursos de rádio dos nós sensores escolhidos para a transmissão de dados. Como o nó fonte tem dois rádios disponíveis, e ele é responsável apenas por transmitir pacotes, então ele deve transmitir os pacotes pelos dois rádios simultaneamente, de forma que os dois rádios estejam ocupados o tempo todo enquanto existem pacotes a serem enviados. No nó de destino, os dois rádios devem estar simultaneamente recebendo pacotes. Enquanto nos nós intermediários, um rádio deve estar ocupado recebendo pacotes e o outro encaminhando pacotes. Além de alternar entre rádios a cada salto, assim como nos protocolos anteriores, utiliza-se a alternância de canais para rádios que transmitem na mesma banda de frequências. São assinalados quatro canais para cada banda. A figura 2 mostra um exemplo de alocação de rádios e canais nos dois caminhos utilizados. Observe que em todos os nós, são utilizados os dois rádios disponíveis.

No nó fonte, os pacotes a serem enviados vão sendo colocados em uma fila. A cada pacote é assinalado um número de sequência para permitir a sua identificação e garantir o recebimento correto no nó de destino. Primeiramente, dois pacotes são retirados da fila, um é enviado por um rádio e o outro enviado pelo outro. Os destinos do próximo salto dessas mensagens são consultados em uma tabela de roteamento que indica dois endereços diferentes para o mesmo destino final, um para cada rádio. Assim que é sinalizado o fim da transmissão de um dos rádios e a transmissão ocorreu com sucesso,

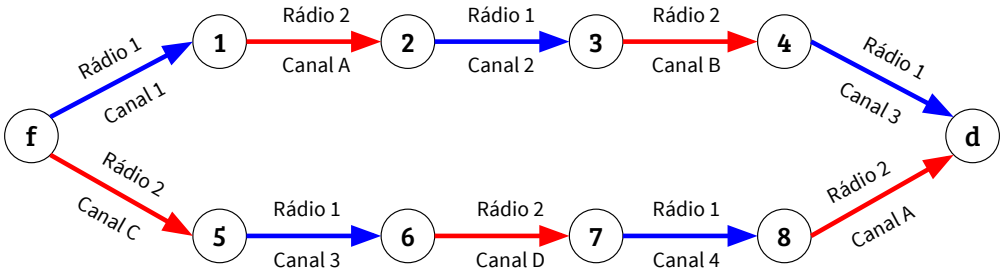


Figura 2. Esquema de alocamento de rádios e canais utilizado no novo protocolo

outro pacote é retirado da fila e transmitido pelo rádio que havia finalizado a transmissão. Como o tempo necessário para a transmissão de um pacote é muito maior do que o tempo necessário para o seu processamento, os dois rádios ficam transmitindo pacotes simultaneamente a maior parte do tempo. O procedimento se repete até que todos os pacotes tenham sido transmitidos.

Em cada um dos nós intermediários nos dois caminhos, ao receber um pacote por um dos rádios, ele consulta a sua tabela de roteamento para determinar o endereço do próximo salto, para o qual ele deve encaminhar essa mensagem. Nos nós intermediários, a tabela de roteamento contém apenas um endereço de encaminhamento para cada destino, pois cada nó intermediário pode fazer parte de apenas um dos caminhos entre o nó fonte e o nó de destino. A mensagem é colocada em uma fila e, sempre que o rádio transmissor fica livre, uma mensagem é retirada da fila e enviada por esse rádio. Novamente, como o tempo de recebimento e envio de pacotes é muito maior do que o tempo de processamento, os dois rádios passam a maior parte do tempo recebendo e transmitindo pacotes simultaneamente.

No nó de destino, os dois rádios são utilizados para receber pacotes. Por um rádio virão pacotes enviados pelo primeiro caminho e pelo outro rádio virão pacotes enviados pelo outro caminho. Os pacotes poderão chegar em ordem diferente da qual foram enviados, por vários motivos: um caminho pode ser mais longo do que o outro, ou um caminho pode ter menos perdas do que o outro. Cada pacote, ao ser recebido pelo nó de destino é sinalizado para a aplicação, juntamente com o seu número de sequência, porém na ordem de chegada. Fica a cargo a aplicação juntar os pacotes recebidos novamente na sequência correta, de acordo com a sua necessidade.

O novo protocolo tenta entregar a maior quantidade de pacotes possível com uma política de melhor esforço. Ele confia em pacotes de confirmação providos pela camada de enlace, e retransmite pacotes que não tenham recebido confirmação até um máximo de cinco retransmissões por pacote. Caso mesmo assim um pacote não consiga ser retransmitido, ele é perdido, e fica também a cargo da aplicação retransmitir ou não os pacotes perdidos de acordo com a sua necessidade. Também é possível configurar o protocolo para desativar o envio e recebimento de pacotes de confirmação da camada de enlace, em caso de necessidade de uma vazão ainda maior ao custo de uma menor confiabilidade.

Para que essa nova abordagem funcione, não basta que sejam utilizados quaisquer dois caminhos disjuntos entre o nó fonte e o nó de destino. A necessidade de alternar os rádios impõe restrições que precisam ser observadas para que tudo funcione. Por isso o

problema de encontrar caminhos em uma rede que sejam compatíveis com o protocolo é discutido na próxima seção.

4. O Problema dos Caminhos

O novo protocolo apresentado utiliza dois caminhos para transmitir dados, porém devido ao fato de cada nó possuir apenas dois rádios distintos que podem operar simultaneamente, esses caminhos devem ser escolhidos de modo que obedeçam a duas restrições. Primeiramente, eles devem ser disjuntos, exceto pelo nó fonte e o nó de destino. Todos os demais nós devem pertencer exclusivamente a um caminho ou outro. Se um nó intermediário for escolhido para os dois caminhos, ele não conseguirá receber e transmitir simultaneamente o fluxo que passa pelos dois, criando um gargalo que faz com que todo o ganho que o protocolo pode oferecer seja perdido.

A segunda restrição é que os dois caminhos devem ter a mesma paridade no seu número de saltos. Precisamos de dois caminhos com número de saltos par ou dois caminhos com número de saltos ímpar. Isso se deve ao fato de que o nó fonte precisa transmitir os pacotes por rádios diferentes e o nó de destino precisa receber os pacotes também por rádios diferentes, para que ambos possam operar simultaneamente. A figura 3 ilustra o porquê dessa restrição. Nela temos uma rede com sete nós, e escolhemos dois pares de caminhos disjuntos diferentes entre o nó fonte s e o nó de destino d. Os caminhos escolhidos em (a) têm paridade diferente. Como o nó fonte precisa enviar por rádios diferentes e os nós intermediários precisam alternar o rádio em que enviam, o nó de destino acaba recebendo os pacotes de ambos os caminhos pelo mesmo rádio, o que será impossível de ocorrer simultaneamente. Já os caminhos escolhidos em (b) têm a mesma paridade. Por isso o nó de destino acaba recebendo pacote pelos dois rádios distintos, o que pode ocorrer ao mesmo tempo.

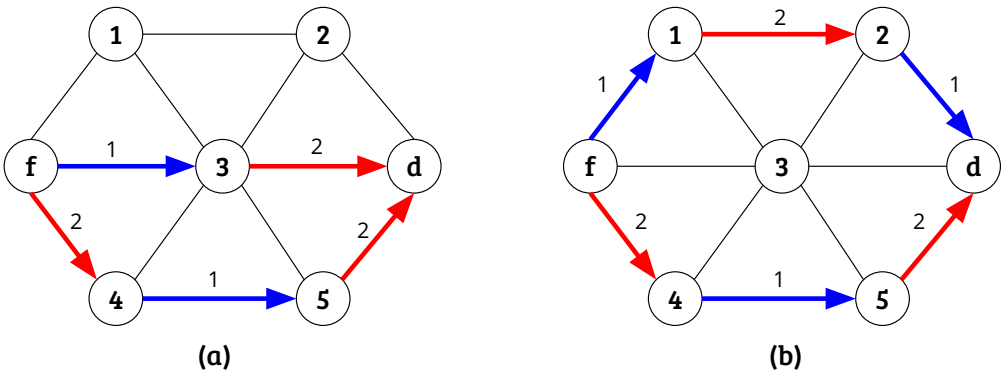


Figura 3. Exemplos de caminhos disjuntos em uma rede. Em (a) com paridades diferentes e em (b) com a mesma paridade.

Um algoritmo de roteamento geralmente estará tentando encontrar o caminho de menor custo entre uma origem e um destino. No caso desse roteamento por dois caminhos distintos, o problema é encontrar o par de caminhos de mesma origem e destino com o menor custo. Caso não houvesse a restrição de que os caminhos tem que ter a mesma paridade, esse seria um problema bem conhecido em teoria dos grafos, que seria facilmente resolvido utilizando o algoritmo de Suurballe [Suurballe and Tarjan 1984]. Porém

tal algoritmo poderia gerar uma solução como a da figura 3 (a), que não serve para ser utilizada com o nosso protocolo.

5. Modelo de Programação Linear Inteira

Para resolver o problema com a restrição de paridade dos caminhos, o modelamos como um problema de programação linear inteira. O modelo completo pode ser lido na figura 4.

$$\text{minimizar} \sum_{(i,j)} c_{ij}^1 x_{ij}^1 + c_{ij}^2 x_{ij}^2$$

sujeito à

$$\text{(i)} \quad \sum_{(i,j)\in S(i)} x_{ij}^1 - \sum_{(j,i)\in E(i)} x_{ji}^2 = \begin{cases} 1, \text{ se } i = f \\ -1, \text{ se } i = d \\ 0, \text{ caso contrário} \end{cases}$$

$$\text{(ii)} \quad \sum_{(i,j)\in S(i)} x_{ij}^2 - \sum_{(j,i)\in E(i)} x_{ji}^1 = \begin{cases} 1, \text{ se } i = f \\ -1, \text{ se } i = d \\ 0, \text{ caso contrário} \end{cases}$$

$$\text{(iii)} \quad \sum_{j\in E(i)} x_{ji}^1 + \sum_{j\in E(i)} x_{ji}^2 \leq 1, \quad \text{se } i \neq d$$

$$\text{(iv)} \quad \sum_{(i,j)\in A} x_{ij}^1 - \sum_{(i,j)\in A} x_{ij}^2 = 0$$

$$\text{(v)} \quad x_{ij}^1, x_{ij}^2 \in \{0, 1\}$$

Figura 4. Modelo de programação linear inteira para resolver o problema de encontrar o menor par de caminhos com a mesma paridade

Representamos a nossa rede como um grafo direcionado $G = (V, A)$. Para cada aresta (i, j) do grafo, associamos dois pesos diferentes, c_{ij}^1 e c_{ij}^2, que representam o custo de enviar uma mensagem do nó i para o nó j. c_{ij}^1 é o custo de enviar a mensagem pelo rádio 1 e c_{ij}^2 o custo de enviar pelo rádio 2. A cada aresta também são associadas duas variáveis binárias, x_{ij}^1 e x_{ij}^2, que assumirão valor 0 se a aresta (i, j) não foi escolhida para nenhum caminho, ou o valor 1, se aquela aresta foi escolhida. x_{ij}^1 representa que o rádio 1 é o rádio escolhido e x_{ij}^2 representa que o rádio 2 foi escolhido. A função objetivo do problema é minimizar o custo total, que é representado pelo somatório do custo de cada rádio multiplicado pela variável x correspondente.

Para modelar as restrições, precisamos fazer mais algumas definições: vamos chamar de f o nó fonte e de d o nó de destino. Para cada nó $i \in V$, vamos definir dois conjuntos de arestas: $S(i)$ é o conjunto de arestas que saem do nó i, e $E(i)$ é o conjunto de arestas que entram no nó i. Vamos então às restrições:

As restrições (i) e (ii) são para garantir a alternância de rádios entre os caminhos escolhidos. Na restrição (i), para cada vértice i, a soma das arestas que saem pelo rádio 1, menos a soma das arestas que entram pelo rádio 2 deve ser 1 se o nó for o nó fonte, -1

se for o nó de destino, ou 0 para todos os outros. Com isso garantimos que o nó fonte não receberá nada pelo rádio 2 e com certeza enviará algo pelo rádio 1. Garantimos que o nó de destino com certeza receberá algo pelo rádio 2 e não enviará nada pelo rádio 1. Para todos os outros nós, se ele recebe algo pelo rádio 2, com certeza ele enviará pelo rádio 1.

Similarmente à restrição anterior, na restrição (ii) para cada vértice i, a soma das arestas que saem pelo rádio 2, menos a soma das arestas que entram pelo rádio 1 deve ser 1 para o nó fonte, -1 para o nó de destino e 0 para todos os outros. Garantimos assim que o nó fonte não receberá nada também pelo rádio 1 e com certeza enviará pelo rádio 2. O nó de destino com certeza receberá algo pelo rádio 1 e não enviará nada pelo rádio 2 e para todos os outros, se o nó recebe algo pelo rádio 1, ele com certeza enviará algo pelo rádio 2.

A restrição (iii) garante que nenhum nó intermediário será escolhido para os dois caminhos, fazendo com que a soma das arestas de entrada em todo nó i, exceto o nó de destino, seja menor ou igual à 1. Como as variáveis são binárias, ou o rádio 1 será escolhido e o rádio 2 não, ou o rádio 2 será escolhido e o rádio 1 não, ou nenhum dos dois será escolhido. Para o nó de destino, de fato, duas arestas de entrada serão escolhidas, uma para cada rádio, como garantem as restrições (i) e (ii). As restrições (i) e (ii) também garantem que não precisamos criar uma restrição para o número de arestas de saída de cada nó, uma vez que ele deverá ser igual ao número de arestas de entrada nos nós intermediários.

Por fim, a restrição (iv) é o que garante que ambos os caminhos terão a mesma paridade. Ela diz que o número de arestas onde o rádio 1 é o escolhido e o número de arestas onde o rádio 2 é escolhido tem que ser iguais. A restrição (v) é para garantir que as variáveis x_{ij}^1 e x_{ij}^2 são binárias.

Com esse modelo de programação linear inteira e dados sobre a qualidade dos enlaces da rede, podemos calcular o menor par de caminhos disjuntos entre um nó fonte e um nó de destino, com caminhos de mesma paridade. Com esses caminhos, podemos utilizar o protocolo proposto nesse trabalho para transmitir dados.

6. Implementação

O protocolo foi implementado usando o TinyOS 2.1.2 para a plataforma Opal [Jurdak et al. 2011]. Essa plataforma dispõe de dois transceptores de rádio 802.15.4 que operam em bandas diferentes: 900 MHz e 2.4 GHz. Os dois rádios compartilham o mesmo barramento SPI, o que cria um gargalo para a transferência de dados entre os rádios e o microcontrolador. Porém, a transferência de dados no barramento é muito mais rápida do que a transmissão de dados pelo rádio. Na plataforma Opal, enviar um pacote SPI para o buffer de transmissão leva menos de 10% do tempo necessário para transmitir o pacote pelo rádio [Ekbatanifard et al. 2013]. Como o protocolo busca manter os rádios sempre ocupados, os dois rádios estarão operando simultaneamente a maior parte do tempo.

A figura 5 mostra uma visão geral da arquitetura implementada para o protocolo, implementada em cima da pilha de protocolos de comunicação para os dois rádios, já disponíveis para TinyOS: o driver dos rádios, e a implementação das camadas MAC e de enlace. Ela oferece opções para configurar a modulação utilizada pelos rádios, a potência

de transmissão, a utilização de verificação de ocupação do canal (CCA, do inglês *Clear Channel Assessment*) e os parâmetros de backoff aleatório utilizados. Ainda há a possibilidade de habilitar ou desabilitar o uso de pacotes de confirmação fornecidos pela camada de enlace.

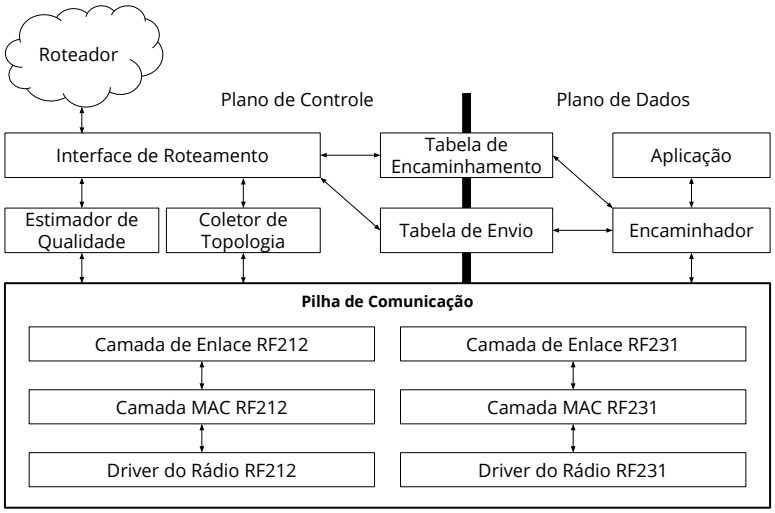


Figura 5. Visão geral da arquitetura implementada

O plano de controle tem a função de encontrar o menor par de caminhos entre um nó fonte e um nó de destino e estabelecer as rotas para os pacotes preenchendo suas tabelas de roteamento. O plano de dados é responsável por enviar e encaminhar os pacotes de acordo com o roteamento pré-definido. Nessa implementação, esse processo se dá em três fases distintas. Primeiro a topologia é coletada, e enviada para o roteador. Segundo, roteamento é calculado offline e as tabelas dos nós sensores escolhidos preenchidas estaticamente. Finalmente, a transmissão dos pacotes é iniciada pelo nó fonte pré-definido no roteamento. A seguir são dados detalhes da implementação de cada uma das etapas.

6.1. Coleta da Topologia

Um estimador de enlace faz com que os nós sensores enviem pacotes periodicamente para que seus vizinhos montam uma tabela que representa a topologia da rede. Ao receber um pacote, o nó sensor adiciona o nó que enviou aquele pacote à sua lista de vizinhos, e passa a contar quantos pacotes recebeu daquele mesmo vizinho. A métrica utilizada para a qualidade do enlace é a razão entre o número de pacotes enviados pelo número de pacotes recebidos entre os nós. Cada nó sensor envia a informação sobre os seus vizinhos pela interface de roteamento, que envia pela porta serial do nó sensor para um computador externo à rede, coletando toda a informação sobre a topologia atual da rede.

6.2. Roteamento

De posse das informações da topologia da rede, o roteamento é feito offline, ou seja, fora da rede de sensores calcula o menor par de caminhos entre determinada fonte e determinado destino na rede. Para isso, utiliza-se o modelo de programação linear inteira descrito na seção 4, implementado utilizando a linguagem GMPL (GNU Mathematical Programming Language) e resolvido utilizando o GLPK (GNU Linear Programming Kit), ferramenta de código aberto para resolver problemas de programação linear. O GLPK levou

em média 107ms para resolver cada instância do problema utilizando o nosso modelo de programação linear inteira, com uma topologia de 100 nós.

A solução encontrada é estaticamente preenchida nas tabelas de roteamento dos nós que fazem parte dos caminhos. Existem duas tabelas: uma tabela de envio e uma tabela de retransmissão. A tabela de envio é utilizada para indicar o próximo salto de cada um dos caminhos para o nó fonte, deve conter duas entradas para cada destino. A tabela de retransmissão é utilizada quando o nó em questão é um nó intermediário no caminho, e pode conter apenas uma entrada para cada destino, que indica o próximo salto. Além do endereço do próximo salto, a tabela indica também o rádio e o canal pela qual a transmissão deve ser feita.

6.3. Transmissão

Depois que as rotas foram estabelecidas, antes de iniciar a transmissão, cada nó altera o canal de operação de seus rádios para os canais indicados na tabela de roteamento. Quando todo o caminho está configurado, o nó fonte inicia a transmissão dos pacotes seguindo o protocolo, descrito na seção 3.

Toda a implementação desse projeto, que inclui o estimador de qualidade, coletor de topologia, o protocolo de transmissão e a modelagem do problema de roteamento usando programação linear são código aberto, e estão disponíveis na internet, em repositório público no GitHub.[1] Além disso também estão disponíveis programas utilizados para realizar os experimentos apresentados na próxima seção.

7. Experimentos e Resultados

Para avaliar o desempenho do protocolo foram realizados experimentos em um testbed de redes de sensores sem fio de larga escala chamado Twonet [Li et al. 2013]. Ele contém 100 nós sensores com dois rádios da plataforma Opal. A interface online provida pelo testbed permite enviar arquivos executáveis compilados para a plataforma Opal, escolher os nós que serão programados, agendar um intervalo de tempo para execução, e como resultado é retornado uma lista de mensagens enviadas para a porta serial de cada nó sensor.

Os experimentos foram feitos para avaliar a vazão e a taxa de entrega de pacotes utilizando o protocolo proposto. Foram feitas várias rodadas de experimentos, e cada rodada consistia nos seguintes passos: coleta da topologia, determinação dos nós de origem e destino, determinação das rotas e medições. O nó fonte e o nó de destino de cada experimento foram escolhidos de acordo com a distância entre eles. Definimos a distância do nó de origem ao destino sendo o número total de saltos dos dois caminhos dividido por 2. Então dois nós em que o par de caminhos entre eles tenham comprimento 5 cada um, estão a uma distância 5. Um par de nós em que os caminhos têm comprimento 4 e 6, também estão a uma distância 5.

Os dois rádios foram configurados para transmitir usando a modulação O-QPSK a 250 kbps e com potência de transmissão de 3 dBm. Nos primeiros experimentos, deixamos habilitadas na camada MAC a verificação da ocupação do canal e os backoffs aleatórios. Depois, essas funções foram desabilitadas a fim de comparar os resultados

[1] `https://github.com/nildo/mpdr`

com os resultados do FastForward. Também realizamos os experimentos com e sem pacotes de confirmação, para analisar o compromisso entre a vazão e a taxa de entrega nos dois casos.

Em cada experimento são enviados 1000 pacotes a partir do nó fonte. Cada pacote tem uma carga útil de 100 bytes, porém o número de bytes efetivamente transmitido pelo rádio para cada pacote foi 127, por causa do custo adicional de vários cabeçalhos de pacote, o que inclui o cabeçalho do protocolo e cabeçalhos da camada de enlace. As métricas utilizadas nos experimentos foram a vazão e a taxa de entrega. Definimos a vazão sendo o número total de bytes recebido pelo nó de destino por segundo, incluindo aqueles não relacionados à carga útil no pacote. Definimos a taxa de entrega como o número de pacotes únicos recebidos pelo nó de destino dividido pelo número total de pacotes enviados pelo nó fonte. Os experimentos foram repetidos 10 vezes para cada instância, e os valores apresentados são a média e o desvio padrão dos resultados obtidos.

A seguir são apresentados gráficos que mostram o desempenho do novo protocolo proposto nesse trabalho que utiliza dois caminhos e comparado com o desempenho da nossa implementação do FastForward. Na figura 6 está o resultado dos testes que foram feitos deixando habilitada a verificação de ocupação do canal (CCA). O gráfico da esquerda mostra a vazão e o gráfico da direita mostra a taxa de entrega. Podemos observar que o novo protocolo conseguiu maior vazão. Em média, o protocolo de dois caminhos conseguiu uma melhora de 60% na vazão nesse cenário. A taxa de entrega do novo protocolo também foi sempre maior ou igual ao FastForward, conseguindo 100% de taxa de entrega em alguns casos devido ao uso de pacotes de confirmação e retransmissões.

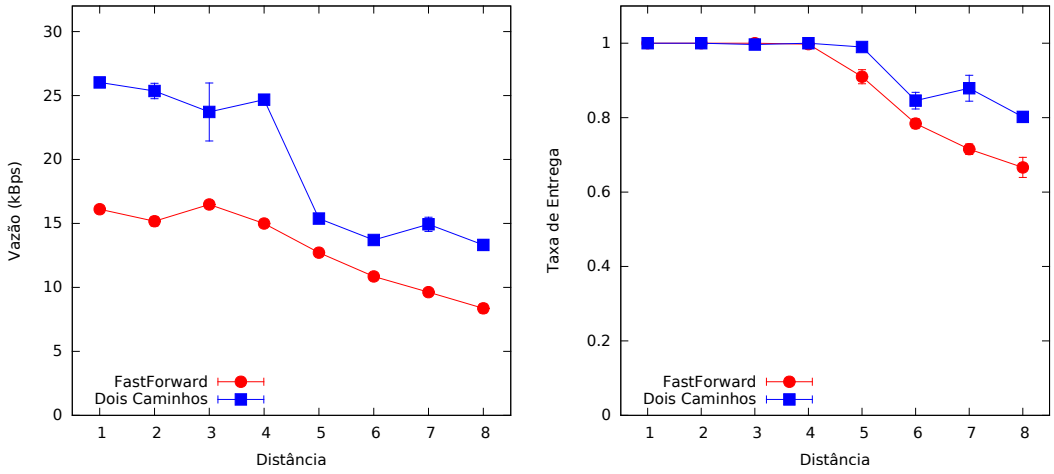


Figura 6. Desempenho com CCA habilitado e com confirmação de mensagens

Ainda observando a figura 6, ambos os protocolos conseguem o máximo de desempenho a até uma distância de 4 entre a fonte e o nó de destino. Como o FastForward usa dois canais em cada rádio, e nossa abordagem utiliza quatro canais, quando o caminho é curto, não há interferência entre canais. A partir de uma distância de 5, passam a ocorrer transmissões no mesmo rádio e no mesmo canal, e essa disputa pelo meio de comunicação faz com que o desempenho dos protocolos piore.

A figura 7 mostra o desempenho dos protocolos quando desabilitamos o CCA na

camada MAC dos rádios. Na prática não é recomendado desabilitar essa função, pois o meio de comunicação pode estar sendo utilizado por várias redes diferentes e uma pode congestionar a outra. O teste foi realizado para avaliar o potencial máximo de transmissão dos protocolos. Nesse cenário também são utilizados os pacotes de confirmação. Podemos observar que nos primeiros casos, obteve-se uma vazão de 50 kBps com o novo protocolo, enquanto o FastForward consegue no máximo 25 kBps, sendo 100% de melhoria, exatamente o limite máximo que se poderia conseguir. Porém o desempenho vai diminuindo a medida em que a distância aumenta. O desempenho nesse cenário cai mais rápido do que o cenário anterior que utilizava o CCA. Nos caminhos mais longos, a vazão é até menor do que antes. Novamente a vazão do novo protocolo foi sempre maior, porém a taxa de entrega entre os dois alternou.

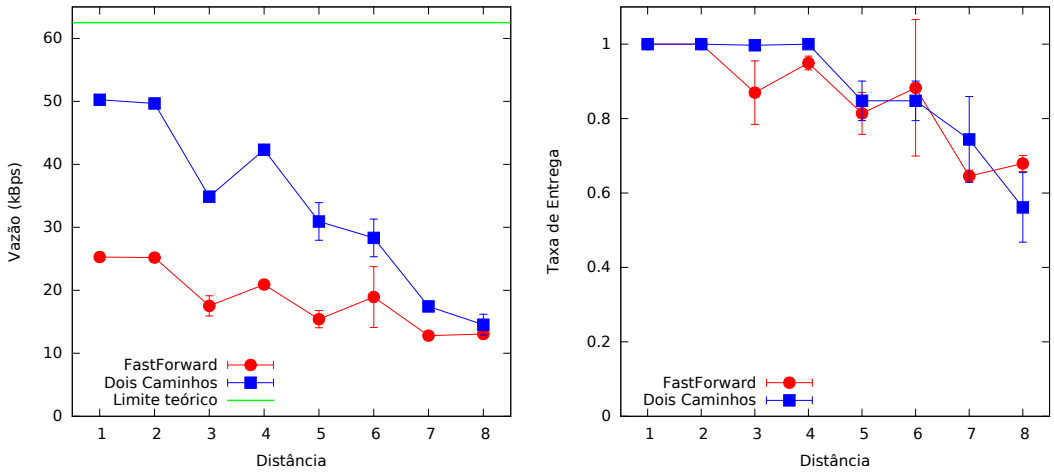


Figura 7. Desempenho com CCA desabilitado e com confirmação de mensagens

Finalmente, a figura 8 representa um cenário onde tanto a verificação de ocupação do canal quanto as mensagens de confirmação foram desativadas. Nesse cenário obtivemos o valor máximo de vazão de 60 kBps nos testes em que a distância era menor ou igual a 4. Esse valor é próximo ao limite teórico máximo de vazão ao utilizar dois rádios. A taxa de transmissão máxima, considerando um ambiente ideal, utilizando apenas um rádio é 250 kbps, ou 31,25 kBps. Então o limite teórico para dois rádios é o dobro, ou 62,5 kBps. Logo, o valor conseguido é 96% da vazão máxima teórica e em um ambiente real. Além disso, esse valor é o dobro do valor conseguido com o FastForward, indicando o máximo aproveitamento dos dois rádios em todos os nós do caminho.

Ainda observando a figura 8, podemos observar uma queda drástica na vazão a partir da distância igual a 5. Essa queda se dá devido à queda na taxa de entrega, que podemos observar no gráfico da direita. Sem CCA, muitos pacotes são perdidos devido à interferência causada entre enlaces que estão compartilhando o mesmo canal, e sem pacotes de confirmação não há a retransmissão de pacotes perdidos. Então apesar do desempenho duas vezes maior para caminhos mais curtos, para caminhos longos o desempenho dos dois protocolos foi praticamente o mesmo, sendo que o FastForward obteve maior taxa de entrega.

Esse último cenário mostra que o protocolo proposto sofre mais com a interferência do que o FastForward, uma vez que ele utiliza mais enlaces. Porém em uma

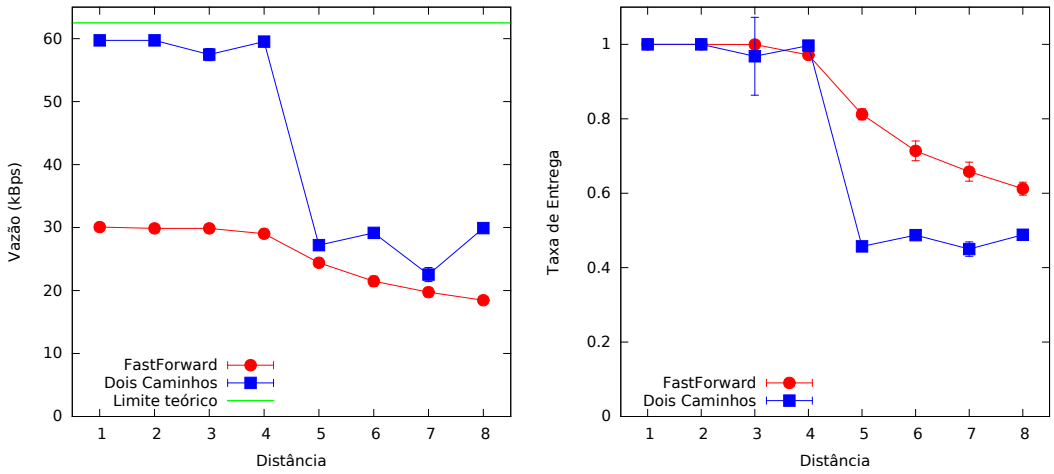


Figura 8. Desempenho com CCA desabilitado e sem confirmação de mensagens

aplicação prática, não seria recomendado desativar a verificação de ocupação do canal, uma vez que é um mecanismo importante para evitar a saturação do meio de comunicação inclusive com outras redes que possam estar presentes no mesmo espaço físico. Como vimos, quando o CCA é habilitado, o desempenho do novo protocolo é melhor do que o do FastForward até para caminhos mais longos.

Uma última questão a ser discutida é em relação ao custo de energia envolvido. Com uma vazão maior, é esperado que o gasto de energia total seja maior. Porém o gasto de energia por byte transmitido se torna menor. Segundo [Jurdak et al. 2011], o Opal consome em média 49 mA de corrente se estiver operando os dois rádios simultaneamente. Como nosso protocolo alcançou uma vazão de 60 kBps, temos um gasto de energético de 0,82 mA/kB. Enquanto com a vazão máxima alcançado pelo FastForward, de 30 kBps, teríamos um gasto energético de 1,64 mA/kB. Logo, nosso protocolo consome metade da energia por byte transmitido em cada nó. Como o número de nós ao utilizar dois caminhos não dobra, pois os nós de origem e destino são sempre os mesmos, o nosso protocolo terá uma eficiência energética total ainda maior do que o FastForward.

8. Conclusão

Nesse artigo foi apresentado um novo protocolo para transferência de dados em massa otimizado para plataformas que utilizam dois rádios. A principal vantagem dessa nova abordagem é a utilização dos dois transceptores disponíveis em cada nó em paralelo, tanto nos nós de origem e destino quanto nos nós intermediários. O protocolo visa atender a demanda por alta vazão em novas aplicações de redes de sensores sem fio que precisam transmitir dados multimídia e em tempo real.

Os experimentos realizados no mundo físico mostram que a abordagem proposta consegue uma vazão de até 60 kBps, o que representa 96% do limite máximo teórico de 62,5 kBps, quando utilizamos dois rádios 802.15.4 utilizando modulação O-QPSK a 250 kbps em paralelo, sem verificação de ocupação do canal. Para uma aplicação que precisa dividir o meio de comunicação e utilizar a verificação de canal ocupado, nosso protocolo consegue uma vazão de até 26 kBps contra 16 kBps do protocolo estado-da-arte FastForward, uma melhora de 60% de desempenho.

Como trabalho futuro, pode-se desenvolver um esquema de roteamento descentralizado para dois caminhos, que possa ser calculado de dentro da própria rede. O problema de encontrar caminhos tem a possibilidade de ser um problema difícil, provavelmente necessitando de boas heurísticas ou soluções aproximadas. Além disso incluir mecanismos que minimizem a interferência cocanal entre os caminhos, uma vez que é o que mais afeta o desempenho do protocolo.

Referências

Akyildiz, I. F. and Vuran, M. C. (2010). *Wireless sensor networks*, volume 4. John Wiley & Sons.

Duquennoy, S., Österlind, F., and Dunkels, A. (2011). Lossy links, low power, high throughput. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 12–25. ACM.

Ekbatanifard, G., Sommer, P., Kusy, B., Iyer, V., and Langendoen, K. (2013). Fast-forward: High-throughput dual-radio streaming. In *2013 IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems*, pages 209–213. IEEE.

Jurdak, R., Klues, K., Kusy, B., Richter, C., Langendoen, K., and Brünig, M. (2011). Opal: A multiradio platform for high throughput wireless sensor networks. *Embedded Systems Letters, IEEE*, 3(4):121–124.

Kim, S., Fonseca, R., Dutta, P., Tavakoli, A., Culler, D., Levis, P., Shenker, S., and Stoica, I. (2007). Flush: a reliable bulk transport protocol for multihop wireless networks. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 351–365. ACM.

Kusy, B., Richter, C., Hu, W., Afanasyev, M., Jurdak, R., Brünig, M., Abbott, D., Huynh, C., and Ostry, D. (2011). Radio diversity for reliable communication in wsns. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 270–281. IEEE.

Li, Q., Han, D., Gnawali, O., Sommer, P., and Kusy, B. (2013). Twonet: Large-scale wireless sensor network testbed with dual-radio nodes. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 89. ACM.

Raman, B., Chebrolu, K., Bijwe, S., and Gabale, V. (2010). Pip: A connection-oriented, multi-hop, multi-channel tdma-based mac for high throughput bulk transfer. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 15–28. ACM.

Suurballe, J. W. and Tarjan, R. E. (1984). A quick method for finding shortest pairs of disjoint paths. *Networks*, 14(2):325–336.

Tavares, R., Vieira, M. A. M., and Vieira, L. F. M. (2016). Flushmf: A transport protocol using multiple frequencies for wireless sensor network. In *Proceedings of the 13th IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS)*. IEEE.

Yick, J., Mukherjee, B., and Ghosal, D. (2008). Wireless sensor network survey. *Computer networks*, 52(12):2292–2330.

Trilha Principal do SBRC 2017
Sessão Técnica 8
Redes Tolerantes a Atrasos

Gerenciamento de Buffer Baseado em Egoísmo para Redes Tolerantes a Atrasos e Desconexões

Camilo B. Souza[1], **Edjair Mota**[1], **Leandro Galvão**[1] **Diogo Soares**[1]

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Avenida General Rodrigo Octávio Jordão Ramos – 3000 – Manaus – AM – Brasil

{camilo.souza,edjair,galvao,diogo.soares}@icomp.ufam.edu.br

Abstract. *The delivery rate in delay tolerant networks is directly impacted by the buffer management algorithm. Recently, by considering the social characteristics of the network node brought a new point of view to the design of such algorithms. This work aims to propose an algorithm that takes into account the selfishness of each node when it has to discard a message from its buffer. A learning machine technique estimates the friendship strength among the nodes. To improve this classification, the proposed algorithm makes use of a training database from an experiment carried out with users from the MIT campus. By means of stochastic simulation, the validation tasks used the Reality and Cambridge traces for the sake of comparison with similar works. The results showed that the proposed algorithm increased the delivery rate while reducing both the average delivery time and the number of hops to deliver the message to the destiny.*

Resumo. *Em redes tolerantes a atrasos e desconexões, a taxa de entrega é diretamente influenciada pelo algoritmo de gerenciamento de buffer. Recentemente, a consideração de características sociais trouxe um novo ângulo de visão no projeto desses algoritmos. O algoritmo apresentado neste trabalho leva em consideração o egoísmo dos nós para decidir que mensagem descartar do buffer. A força da amizade entre dois nós, preponderante para essa decisão, é classificada utilizando-se uma técnica de aprendizagem de máquina. Para melhorar a qualidade dessa classificação, utiliza-se como dados de treinamento características de amizades extraídas de um experimento realizado por pesquisadores do MIT com usuários do campus universitário. Utilizou-se simulação estocástica baseada nos traces Reality e Cambridge para efeito de comparação com resultados de trabalhos similares. Os resultados obtidos mostram que o algoritmo proposto contribui para o aumento da taxa de entrega na rede, diminuição do tempo médio de entrega, além de uma razoável média de saltos para que uma mensagem alcance o seu destinatário.*

1. Introdução

Redes tolerantes a atrasos e desconexões (DTN) compartilham de características comuns tais como longos atrasos e frequentes desconexões causados principalmente pelo mau funcionamento dos protocolos que atuam na pilha TCP/IP em decorrência da ausência de uma ou mais premissas básicas necessárias para o seu bom funcionamento [de Oliveira et al. 2007].

A camada de agregação proposta por Fall et al. [Fall 2003], posicionada entre a camada de aplicação e a camada de transporte do referido modelo, permite a implementação do paradigma armazena-carrega-encaminha. Na ausência de um caminho fim-a-fim entre o nó origem e nó destino, é possível armazenar de forma persistente uma mensagem [de Oliveira et al. 2007]. No entanto, a combinacão desse paradigma de rede com algoritmos de roteamento baseados em replicações de mensagens, pode causar a situação denominada de *overflow* [Naves et al. 2012], e uma ou mais mensagens devem ser escolhidas para serem descartadas do *buffer* dos nós .

Algoritmos de gerenciamento de *buffer* para DTN auxiliam na decisão de que mensagem(ns) retirar em casos de *overflow* do *buffer* dos nós. Na literatura encontram-se diversas propostas de algoritmos de gerenciamento de *buffer* para DTN. Tang et al. [Tang et al. 2012] propuseram uma classificação para os algoritmos de gerenciamento de *buffer* existentes dividindo-os em duas categorias: algoritmos que não usam informações da rede e algoritmos que usam informações da rede. Recentemente, a consideração de características sociais forneceu um novo ângulo de visão no projeto de algoritmos para DTN [Xia et al. 2015], surgindo, assim, uma terceira categoria de algoritmos de gerenciamento de *buffer* baseada em características sociais. Dentre as características sociais mais abordadas estão o egoísmo, a amizade, a centralidade, a popularidade, dentre outras.

Este trabalho apresenta o projeto e a avaliação de desempenho de um algoritmo de gerenciamento de *buffer* para DTN, aqui denominado *Selfish Drop-Based* (SDB), que leva em consideração o egoísmo dos nós que compõem a rede. Para isso, utilizou-se a classificação do egoísmo dos nós realizada por [Zhu et al. 2013], a qual divide o egoísmo de um nó em social ou individual. O nó egoísta individual é aquele que não aceita mensagens destinadas a outros nós; está interessado somente na sua participação dentro da rede. O nó egoísta social aceita mensagens destinadas a outros nós, porém somente àqueles com quem ele tem alguma relação social.

Para realizar o gerenciamento do *buffer* dos nós, o algoritmo SDB classifica os nós quanto ao seu nível de egoísmo e escolhe que mensagem deve ser descartada. Se o egoísmo do nó for classificado como individual, o nó descarta a mensagem candidata a entrar no *buffer*, se for classificado como social, o nó prioriza aquelas destinadas a outros nós com quem ele tem uma amizade forte. Dependendo das limitações de seus recursos (nível de bateria, por exemplo), o nó pode tornar-se egoísta. Nó não egoísta e sem limitações de recursos, executa o descarte levando em consideração a força da amizade entre os nós.

As principais contribuições deste trabalho são:

- considerar o egoísmo no projeto de algoritmos para o gerenciamento do *buffer* dos nós em DTN – uma vez que DTN podem ser formadas por humanos equipados com dispositivos móveis, e o seu comportamento influencia na maneira como são gerenciados os recursos dos dispositivos que compõem a rede, é de suma importância levar em consideração o egoísmo no projeto de algoritmos de gerenciamento de *buffer* para DTN. O algoritmo proposto nesse trabalho visa preencher essa lacuna observada na literatura.
- considerar que nós não egoístas podem se tornar egoístas por conta da criticidade dos seus recursos – os algoritmos para DTN que consideram o egoísmo dos nós, consideram que os nós não egoístas não serão egoístas em nenhuma situação.

Nesse trabalho, considera-se que os nós não egoístas podem tornar-sem egoístas individuais em situações de criticidade dos seus recursos, para garantir o uso do seu dispositivo em favor de si mesmo.

- classificar a força da amizade entre os nós usando técnicas de aprendizagem de máquina – os algoritmos para DTN que consideram a amizade entre os nós, utilizam determinadas métricas para a definição da força da amizade entre os nós. O algoritmo proposto no presente trabalho utiliza uma abordagem diferenciada para classificar a força da amizade entre os nós, a qual inclui o uso de um algoritmo de aprendizagem de máquina combinado com dados de amizades extraídas de um experimento realizado por pesquisadores do MIT com usuários do campus universitário.

O restante desse trabalho está organizado da seguinte forma: na seção 2 são apresentados os principais trabalhos relacionados ao tema abordado nesse trabalho, enquanto que na seção 3 maiores detalhes sobre o algoritmo proposto são dados. A seção 4 apresenta detalhes sobre a metodologia de avaliação, enquanto que a seção 5 aborda os resultados obtidos nas avaliações realizadas. Finalmente, na seção 6 apresentam-se as conclusões e os trabalhos futuros.

2. Trabalhos relacionados

Em DTN, a taxa de entrega é diretamente influenciada pelo algoritmo de gerenciamento de *buffer* [Naves et al. 2012], visto que a decisão de descartar ou não uma determinada mensagem pode possibilitar ou impossibilitar sua entrega em contatos posteriores. Os algoritmos que não utilizam informações da rede, geralmente realizam apenas uma escolha simples para tomar a decisão de descarte. Esses algoritmos utilizam informações das mensagens como por exemplo o tempo de vida da mensagem (TTL) e o tamanho da mensagem. Exemplos de algoritmos pertencentes a esta classe são as seguintes: descarte aleatório [Rashid et al. 2011], descarte da mensagem menos recentemente recebida [Lindgren and Phanse 2006], descarte da mensagem mais antiga no *buffer* [Rashid et al. 2011], descarte da última mensagem na fila [Lindgren and Phanse 2006], descarte da primeira mensagem na fila [Rashid et al. 2011], descarte da mensagem mais nova [Rashid et al. 2011] e descarte da mensagem de maior tamanho [Lindgren and Phanse 2006].

Os algoritmos que utilizam informações da rede, por sua vez, utilizam uma abordagem mais sofisticada levando em consideração informações como o histórico de encontros dos nós da rede [Krifa et al. 2008], a frequência de encontros entre os nós da rede [Tang et al. 2012], a probabilidade de entrega e a estimativa de disseminação de uma mensagem na rede [Naves et al. 2012]. No entanto, mais recentemente a consideração de características sociais trouxe um novo ângulo de visão no projeto de algoritmos para DTN e uma terceira classe de algoritmos de gerenciamento de *buffer* surgiu. Esses algoritmos fazem uso em suas estratégias de características como amizade, egoísmo, centralidade, popularidade, similaridade, entre outras.

Em [Souza et al. 2014], Souza et al. propõem um algoritmo denominado *Drop Less Known* (DLK) para o gerenciamento de *buffer* dos nós de redes DTN. Esse algoritmo introduz uma métrica denominada força da relação social, a qual considera o quão forte são os laços sociais entre dois nós (forte, médio, fraco). Para classificar a relação

social entre um par de nós, os autores utilizaram a quantidade de contatos e um log de ligações entre os nós disponibilizados em um conjunto de dados denominado *Reality*. Os experimentos realizados mostraram que o algoritmo DLK contribuiu para um aumento na taxa de entrega e houve diminuição razoável do atraso médio de entrega de mensagens na rede.

No presente trabalho, os dados do trace *Reality* são usados como dados de treinamento para que um algoritmo de aprendizagem de máquina classifique a força da amizade entre dois nós (forte, fraco). Uma outra diferença deste trabalho para [Souza et al. 2014] está na forma como é feita a classificação da relação social, visto que o algoritmo DLK atribui de maneira estática (antes da simulação iniciar) a força da relação social e essa relação não muda, enquanto que a classificação da relação social (amizade) feita no presente trabalho é feita de maneira dinâmica e por esse motivo pode se alterar durante a simulação, representando assim um comportamento mais realista.

Uma outra característica social abordada em algoritmos de gerenciamento de *buffer* para DTN é a centralidade, que pode ser definida como a quantidade de interação de um nó com os demais nós da rede. Quanto maior for a centralidade de um nó, pode-se concluir que maior interação com outros nós ele tem. Dessa forma, [Settawatcharawanit et al. 2013] propõem um algoritmo para o gerenciamento de *buffer* baseado na centralidade dos nós dentro de comunidades locais. Para calcular a centralidade de um nó dentro de uma comunidade local, o algoritmo analisa as médias de encontros entre os nós em determinadas janelas de tempo. Quando ocorre o *overflow*, o nó procura em seu *buffer* uma mensagem destinada a um nó que não pertença a mesma comunidade que a sua. Se todas as mensagens armazenadas no *buffer* são destinadas a nós da mesma comunidade, a mensagem mais antiga é retirada do *buffer*. Assim como esses autores, no presente trabalho também considera-se a quantidade de encontros para determinar a relação social (amizade), no entanto, consideram-se outras informações que contribuem para aumentar a qualidade da classificação da relação social realizada, a saber: histórico de trocas de mensagens de texto e ligações e a duração dos contatos.

Apesar de haver uma quantidade significativa de trabalhos e de consenso sobre a arquitetura geral das redes DTN [Cerf et al. 2007], ainda não existe um consenso sobre algoritmos de gerência de *buffer* para as mesmas. De acordo com [Tang et al. 2012], os algoritmos encontrados na literatura podem melhorar o desempenho da rede em um determinado grau, mas todos têm suas limitações. Nesse intuito, este trabalho apresenta uma política de gerência de *buffer* que utiliza uma característica social ainda não abordada nesse tema, trata-se do egoísmo entre os nós. A seção a seguir apresenta maiores detalhes sobre o algoritmo proposto.

3. Selfish Drop-Based Algorithm

O algoritmo proposto no presente trabalho leva em consideração o egoísmo dos nós para tomar a decisão de que mensagem descartar do *buffer*. As principais motivações para se gerenciar o *buffer* dos nós utilizando essa característica como base são:

(i) tornar a decisão do descarte de mensagens mais realista - de acordo com [Li et al. 2010], a maioria dos algoritmos em DTN considera que os usuários estão dispostos a carregar e encaminhar mensagens uns para os outros. No entanto,

no mundo real as pessoas são socialmente egoístas e tendem a carregar mensagens somente para aqueles usuários com quem eles tem alguma relação social, e a vontade de carregar uma mensagem destinada a outra pessoa depende da força da relação social entre eles. Dessa forma, como no gerenciamento de *buffer* uma ou mais mensagens devem ser descartadas em caso de *overflow*, é razoável supor que usuários reais, se questionados sobre que mensagens descartar do *buffer* do seu dispositivo, escolheriam aquelas destinadas a usuários com quem a relação social entre eles é mais fraca ou inexistente. Assim, o algoritmo SDB é uma interessante alternativa para atender esse fato.

(ii) aumentar a taxa de entrega através do egoísmo social - de acordo com [Zhu et al. 2013], o egoísmo social pode ser definido como o tipo de egoísmo em que um nó aceita participar das ações da rede encaminhando mensagens para outros nós, no entanto só aceita carregar mensagens destinadas a outros nós com quem tem certa relação social. Baseado nesse fato, chamando-se de círculo social o grupo de nós com quem se tem certa relação social, é razoável supor que os nós que pertencem ao círculo social de um determinado nó, são também aqueles com quem ele se encontra com uma determinada frequência. Como em DTN a oportunidade de contato é de suma importância para que dois nós se comuniquem, pode-se concluir que quanto maior for a frequência de contatos entre dois nós, maior será a quantidade de dados trocados por eles. Dessa forma, como as oportunidades de contatos são maiores, consequentemente a taxa de entrega de mensagens tende a aumentar.

O algoritmo 1 apresenta o pseudocódigo do funcionamento geral do algoritmo SDB. Basicamente, em uma oportunidade de contato, o algoritmo SDB inicia sua operação verificando se o nó candidato a receber mensagens é egoísta ou não egoísta, e, dependendo dessa verificação incial o algoritmo irá executar um conjunto de operações diferentes. Para melhor entendimento do leitor, as explicações sobre o funcionamento do algoritmo SDB foram divididas em duas subseções, uma que aborda a operação do algoritmo SDB caso o nó candidato a receber a mensagem seja egoísta (chamada de Procedimento 1) e outra que aborda a situação em que o nó candidato é não egoísta (chamada de Procedimento 2).

Algoritmo 1 *Selfish Drop-Based Algorithm*

```
1: procedure SDB(noRecebedor, buffer[])
2:     noRecebedor.tipo ← determinaTipo()
3:     if noRecebedor.tipo ==egoista then procedimento1()
4:     elseprocedimento2()
5:     end if
6: end procedure
```

3.1. Procedimento 1: candidato a receber a mensagem é egoísta

O pseudocódigo 2 apresenta maiores detalhes sobre o funcionamento do algoritmo SDB caso o nó seja identificado como egoísta. Se o nó candidato a receber a mensagem for egoísta, o algoritmo SDB continua as verificações analisando se o tipo de egoísmo do nó candidato a receber a mensagem é individual ou social. Caso seja individual, o algoritmo

SDB descarta a mensagem candidata a entrar no *buffer*. Caso seja social, o algoritmo SDB descarta a mensagem destinada ao nó com quem se tem a relação social mais fraca. Como dito anteriormente, a relação social escolhida para auxiliar na tomada de decisão pelo algoritmo SDB foi a amizade entre os nós, que, no presente trabalho pode ser classificada como fraca ou forte. É interessante destacar ainda que para fazer a classificação da força da amizade, o algoritmo SDB utiliza uma técnica de aprendizagem de máquina denominada Naive Bayes e uma base de dados retirados de um experimento real denominado *Reality* [Eagle and Pentland 2006]. Maiores detalhes sobre essa funcionalidade são dadas a seguir.

3.1.1. Classificação da amizade

Para classificar a força da amizade entre um par de nós na rede, o algoritmo SDB utiliza um algoritmo de aprendizagem de máquina denominado Naive Bayes [Gama and teorema de Bayes 2012, Lewis 1998]. Para escolher-se o algoritmo de aprendizagem de máquina observou-se em outros trabalhos da literatura o desempenho dos principais algoritmos testados, detectando-se que em vários deles o algoritmo escolhido obteve o melhor desempenho na tarefa de classificação [Gama and teorema de Bayes 2012, Lewis 1998].

O algoritmo Naive Bayes é considerado um classificador supervisionado, ou seja, utiliza instâncias juntamente com seus rótulos para classificar novas instâncias. Nesse contexto, uma instância é um exemplo conhecido de valores para os atributos ou características que ajudam na classificação dos rótulos, os quais se tratam de um valor para a classe que se deseja classificar. Por exemplo, supondo que os atributos de interesse no problema em questão sejam o *Round Trip Time*(RTT), o tamanho da fila dos roteadores e a largura de banda disponível. Considerando ainda que os valores de RTT variem entre 50 a 200ms, o tamanho da fila de 0 a 100 , a largura de banda disponível de 0 a 100 e que no problema existam dois rótulos (classes) possíveis : 1 (presença de congestionamento) e 2 (ausência de congestionamento). Nesse contexto, a tupla de valores (75, 50, 80, presença de congestionamento) é considerada um exemplo já conhecido (instância) dentro do problema. De maneira geral, em seu funcionamento, o algoritmo Naive-Bayes, basicamente analisa os valores para os atributos de um exemplo desconhecido e tenta predizer a que classe estes pertencem, baseando-se no histórico de probabilidades para exemplos conhecidos (base de treino).

Algoritmo 2 *Procedimento 1*

1: **procedure** PROCEDIMENTO1($noRecebedor, \textbf{\textit{buffer}}[], M1$)
2: $noRecebedor.tipoEgoismo \leftarrow determinaTipo()$
3: **if** $noRecebedor.tipoEgoismo ==$"individual" **then** $descartarM1$
4: **else**$verificarAmizade()$
5: **end if**
6: **end procedure**

O algoritmo Naive Bayes é baseado no Teorema de Bayes, que é utilizado para o cálculo das probabilidades necessárias para a classificação de uma nova instância. No contexto de aprendizagem de máquina, pode-se definir o Teorema de Bayes da seguinte

forma: dada uma instância desconhecida *A = (a1, a2, a3.. an)*, onde *a1, a2, a3,...an* são valores dos atributos de uma instância, deseja-se predizer sua classe. Para realizar a escolha da classe, o Teorema de Bayes utiliza a seguinte Fórmula:

$$P(Classe|A) = \frac{P(A|Classe) \times P(Classe)}{P(A)} \quad (1)$$

Para cada atributo da instância *A = (a1, a2, a3.. an)*, pode-se reescrever:

$$P(Classe|a1, ..., an) = \frac{P(a1, ..., an|Classe) \times P(Classe)}{P(a1, ..., an)} \quad (2)$$

Em termos gerais, a equação (2) representa a probabilidade de a nova instância ser de uma determinada classe em termos dos valores de cada um dos atributos *A = (a1, a2, a3.. an)*. Além disso, ao observar-se a Equação (2), percebe-se que o denominador é uma constante, sendo assim possível anulá-lo do Teorema de Bayes. Com isso, o Teorema de Bayes fica resumido a:

$$argmax P(a1, ...an|Classe) \times P(Classe) \quad (3)$$

O Teorema de Bayes faz a suposição de que os atributos de uma instância são estatisticamente independentes, ou seja, um determinado valor x para um atributo não implica diretamente em um valor y para outro atributo. No entanto, na maioria das vezes, a suposição de independência entre atributos acaba se tornando falsa. Apesar disso, o algoritmo Naive Bayes produz resultados bastante satisfatórios em diversos problemas reais, ou seja, consegue predizer de maneira correta a classe de uma determinada instância desconhecida [Domingos and Pazzani 1997].

No presente trabalho, o problema para o qual utilizou-se o algoritmo Naive Bayes como classificador pode ser definido da seguinte forma:

- **Tarefa realizada**: classificar a força da amizade entre dois nós em uma rede. As classes possíveis são Amizade Forte ou Amizade Fraca.
- **Experiência de treinamento**: uma base de dados derivada de um experimento em que a força da amizade entre os nós participantes já são conhecidas e definidas como fortes ou fracas.
- **Atributos utilizados como base para a classificação**: quantidade de contatos, duração dos contatos, log de ligações, log de mensagens de texto.

Os atributos utilizados como base para a classificação podem ser descritos da seguinte forma:

- **quantidade de contatos** – o total de encontros entre dois nós em uma semana.
- **duração de contatos** – tempo médio de duração dos encontros em uma semana.
- **log de ligações** – quantidade de ligações trocadas entre dois nós em uma semana.
- **log de mensagens de texto** – quantidade de mensagens de texto trocadas entre dois nós em uma semana.

A base de treino utilizada para classificar novas instâncias foi extraída de um experimento real realizado por pesquisadores do MIT denominado *Reality Mining* [Pentland et al. 2009]. O projeto foi realizado durante seis meses com 97 pessoas equipadas com dispositivos móveis com a interface Bluetooth ativada. Como resultado desse experimento, foi disponibilizado para a comunidade científica diversas informações referentes ao uso dos dispositivos móveis pelas pessoas, tais como conectividade, proximidade, localização e outras atividades dos usuários.

Um dos principais motivos para usar-se essa base de dados no presente trabalho é o log de ligações e de mensagens de texto dos usuários que participaram do experimento. Como citado anteriormente, essa informação é usada como base para classificar a força da amizade entre dois nós na rede. Um outro motivo importante para utilização dessa base de dados no presente trabalho são as respostas para um questionário aplicado aos participantes. Dentre outras perguntas, neste questionário existia uma questão com o seguinte conteúdo: "essa pessoa pertence ao seu ciclo de amizades?". As respostas possíveis eram sim ou não. Dessa forma, para criar-se a base de treino utilizada no presente trabalho, foram extraídas do conjunto de dados *Reality* as seguintes informações sobre a relação entre um par de nós A e B: log de informações e mensagens de texto, quantidade e duração de contatos entre eles e qual a relação entre eles (se são amigos ou não).

3.2. Procedimento 2: candidato a receber a mensagem é não egoísta

O pseudocódigo 3 apresenta maiores detalhes sobre o funcionamento do algoritmo SDB caso o nó seja identificado como não egoísta. Se o nó candidato a receber mensagens é não egoísta, o algoritmo SDB avalia se o mesmo pode se tornar egoísta por conta das limitações dos seus recursos. No presente trabalho, levou-se em consideração o nível de energia como critério para avaliar se um nó não egoísta tornou-se egoísta. A ideia por trás dessa funcionalidade do algoritmo surgiu da hipótese de que em determinadas situações, usuários reais desejariam não participar da rede por conta do nível crítico de recursos do seu dispositivo.

Algoritmo 3 *Procedimento 2*

1: **procedure** PROCEDIMENTO2($noRecebedor$, ***buffer***$[], M1$)
2: $noRecebedor.nivelBateria \leftarrow obtemNivelBateria()$
3: **if** $noRecebedor.nivelBateria <=$"20%" **then** $descartarM1$
4: **else**$verificarAmizade()$
5: **end if**
6: **end procedure**

Essa hipótese foi validada através da execução de um pequeno experimento com objetivo de avaliar duas questões, a saber: (i) como usuários reais se comportariam em situações de participação da rede com nível de energia crítico e (ii) a partir de que nível de energia os usuários considerariam participar da rede somente em favor de si próprio (não aceitando mensagens para os demais). Para responder essas questões, executou-se um experimento através da aplicação de um questionário contendo duas questões com a participação de 100 pessoas. Os participantes do experimento foram pessoas de diversos cursos diferentes da Universidade Federal do Amazonas. Além disso, o questionário permaneceu disponível para preenchimento durante um mês. O conteúdo do questionário é apresentado abaixo:

> "Imagine que você está em um lugar isolado onde a comunicação só possa ser feita por um aplicativo de troca de mensagens chamado MSG1. Porém, diferente de outros aplicativos , no MSG1 para você se comunicar com outra pessoa você precisa "parear"seu dispositivo com o da outra pessoa via Wifi ou Bluetooth enviando antes um convite para se conectar com aquela pessoa. Neste cenário, responda: "

- Pergunta 1 – Se uma pessoa envia uma solicitação de conexão, porém a bateria do seu dispositivo está em um nível crítico, o que você faria? (a) aceitaria o convite (b) não aceitaria o convite (c) esperaria recarregar a bateria para depois comunicar com a pessoa.
- Pergunta 2 – A partir de qual nível você considera que a bateria do seu dispositivo está em um nível crítico? (a) abaixo de 10% (b)de 10% a 20% (b)21% a 30%

A Tabela 1 apresenta os resultados obtidos nesse experimento. Como pode-se observar, 55% das pessoas responderam que "esperariam recarregar a bateria para depois comunicar com a pessoa". Considerando-se que apenas 10% das pessoas respondeu que aceitaria o convite para se comunicar, pode-se concluir que de certa forma, 90% das pessoas deixaria para se comunicar posteriormente. A conclusão que pode ser tirada desse experimento é que nessa situação de criticidade no nível de bateria dos dispositivos dos nós, o nó candidato a receber uma mensagem deve decidir não recebê-la por que deseja economizar energia em favor da sua própria participação na rede. Dessa forma, é razoável supor que nesse caso nós não egoístas tornem-se temporariamente egoístas do tipo individual e portanto devem descartar a mensagen candidata a entrar no *buffer*. Além disso, nesse trabalho considera-se um dispositivo em nível crítico de bateria quando o seu valor está abaixo de 20%, baseando-se nas respostas obtidas no questionário.

Tabela 1. Respostas em % para as perguntas do questionário aplicado no experimento

Pergunta	(a)	(b)	(c)
1	10%	35%	55%
2	28%	67%	5%

Para efeito de simulação, no presente trabalho assumiu-se que 40% dos nós participantes da simulação são egoístas (20 % individual e 20%social), enquanto que 60% dos nós são não egoístas.

4. Metodologia

Para validar a proposta de gerência de *buffer* baseada em egoísmo para redes DTN, utilizou-se a técnica da simulação estocástica. Nesse trabalho, implementou-se um modelo de simulação que inclui o algoritmo *Selfish Drop-Based Algorithm* e outros algoritmos disponíveis na literatura, permitindo assim quantificar o desempenho do algoritmo proposto. Os algoritmos escolhidos para comparação foram os algoritmos *Drop Less Known* [Souza et al. 2014], e o algoritmo proposto por [Settawatcharawanit et al. 2013], doravante chamado de *Drop Centrality-Based* (DCB). A motivação para a escolha desses algoritmos é a similaridade das suas propostas que também são baseadas em características sociais, da mesma forma que o algoritmo proposto no presente trabalho. O cenário

de simulação foi implementado no simulador *Opportunistic Network Environment* (The ONE) [Keränen et al. 2009]. O simulador The ONE foi selecionado levando em consideração os demais trabalhos na literatura que trabalharam com simulação, os quais em sua grande maioria utilizaram o referido simulador para a implementação do seu modelo.

Para simular a mobilidade dos nós, utilizou-se traces de mobilidade real escolhendo-se dois traces bem conhecidos disponíveis na literatura, chamados de *Cambridge* e *Reality*. A escolha de traces de mobilidade real em vez de mobilidade sintética deu-se principalmente pelos inconvenientes citados nesta última forma de mobilidade em [Yoon et al. 2003, Resta and Santi 2002]. A Tabela 2 apresenta maiores detalhes sobre os traces utilizados neste trabalho. Utilizou-se ainda como algoritmo de roteamento o protocolo Epidêmico [Vahdat et al. 2000]. A principal motivação para utilização desse protocolo é a maneira como o mesmo replica as mensagens na rede, favorecendo o *overflow* e consequentemente necessitando de um bom gerenciamento de *buffer*.

Tabela 2. Traces baseados em movimentação humana real

Trace	Dispositivo	Tipo de rede	Número de Nós	Duração	Granularidade	Número de Contatos
Cambridge	*iMotes*	*Bluetooth*	36	11 dias	100 segundos	10873
Reality	*Telefone*	*Bluetooth*	97	246 dias	300 segundos	54667

Segundo [Fathima e Wahidabanu, 2011], o desempenho de uma rede DTN é medido em termos da razão da taxa de entrega e do atraso médio de entrega. Dessa forma, realizou-se uma comparação do desempenho de várias políticas de gerência de *buffer* avaliando-se três métricas: taxa de entrega, atraso médio e média de saltos.

Para simular o consumo de energia dos nós, adotou-se o modelo de energia proposto em [Silva et al. 2012]. Nesse modelo, o consumo de energia é classificado em cinco estados

- **Desligado** – não há consumo de energia, simula a situação em que a interface de rede está desligada.
- **Inativo** – consumo de energia reduzido, simula a situação em que a interface de rede está em modo de espera.
- **Scan** – o nó consome energia para descobrir outros nós vizinhos.
- **Transmitindo** – o nó consome energia enquanto envia uma mensagem.
- **Recebendo** – o nó consome energia enquanto está recebendo uma mensagem.

Nas simulações realizadas, assumiu-se que os nós incialmente têm um nível máximo de energia. Para efeito de simulação, considerou-se que o nível de energia dos nós é medido em unidades e inicialmente cada um deles tem 500 unidades de energia (nível máximo de energia). Assumiu-se também que os dispositivos são recarregados uma vez a cada 24 horas. O consumo de energia do dispositivo depende do estado e do número de operações usando a sua interface de rede. Por exemplo: se um nó está transmitindo, recebendo ou procurando vizinhos, assumiu-se que são reduzidas 25 unidades para cada envio/recebimento de mensagem e/ou para cada scaneamento realizado. Se o nó está no modo desligado ou inativo, assumiu-se que não existe consumo de energia.

5. Resultados obtidos

Nesta seção são apresentados os resultados obtidos ao executar o projeto de experimentos definido na seção anterior.

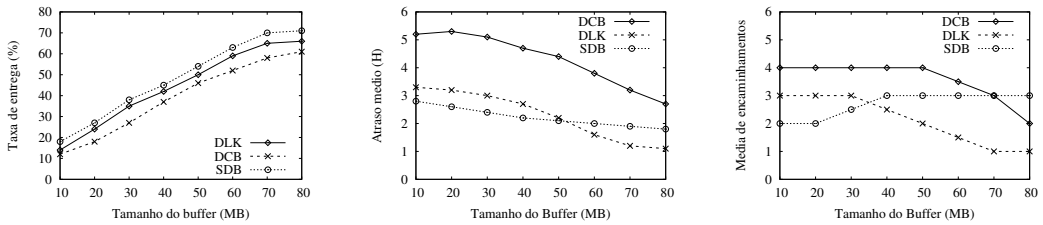


Figura 1. *Resultados para o trace de Cambridge*

5.1. Taxa de entrega

A taxa de entrega pode ser definida como a razão entre a quantidade de mensagens entregues pela quantidade de mensagens criadas na rede. As Figuras 1.(a) e 2.(a) apresentam os resultados obtidos para a métrica taxa de entrega nos cenários de mobilidade Cambridge e *Reality*. Com relação à métrica taxa de entrega, a hipótese levantada era a de que a consideração do egoísmo no gerenciamento do *buffer* dos nós poderia causar um impacto positivo no que diz respeito à quantidade de mensagens entregues na rede. Como pode-se observar nos resultados para à métrica taxa de entrega, o algoritmo SDB obteve o melhor resultado em ambos os cenários, entregando mais mensagens que as demais políticas testadas, validando a hipótese levantada.

A justificativa para esse desempenho está ligada diretamente à consideração da força da amizade como um subcritério para descartar mensagens do *buffer* dos nós nos casos de nós egoístas sociais e não egoístas. Em ambos os casos, a escolha de que mensagem deverá ser retirada do *buffer* é baseada na força da amizade entre o nó que está tomando a decisão e os destinatários das mensagens armazenadas no *buffer*. Como citado anteriormente, os nós que são amigos tendem a estarem em contato com uma frequência maior do que nós que não são amigos, favorecendo assim que mensagens sejam entregues entre si.

5.2. Atraso médio

O atraso médio pode ser definido como o tempo médio necessário para que uma mensagem alcance o seu destinatário. As Figuras 1.(b) e 2.(b) apresentam os resultados obtidos para a métrica atraso médio nos cenários de mobilidade Cambridge e *Reality*. Com relação à métrica atraso médio, a hipótese levantada era a de que a utilização do egoísmo no gerenciamento do *buffer* contribuiria para o aumento do tempo médio de entrega de uma mensagem. De acordo com os resultados apresentados para esta métrica, pode-se perceber que o algoritmo SDB alcançou o melhor desempenho no cenário de Cambridge para tamanhos de *buffer* menores que 60 MB, sendo superado pelo algoritmo DLK quando o tamanho de *buffer* foi maior ou igual a 60 MB. Porém, no cenário *Reality* o algoritmo SDB obteve o melhor resultado apresentando uma diferença de até duas horas a menos de atraso médio que os demais algoritmos testados. Portanto, a hipótese levantada mostrou-se falsa.

Fazendo-se uma análise do comportamento dos resultados obtidos pelo algoritmo SDB, percebe-se uma leve variação no seu desempenho em função do aumento do espaço em *buffer*. Por exemplo, no cenário de Cambridge a diferença entre o resultado para tamanho de *buffer* igual a 10 MB e o de 80 MB foi de apenas 1 hora. A partir dessa observação, é razoável supor que o atraso médio do algoritmo SDB sofre um leve impacto

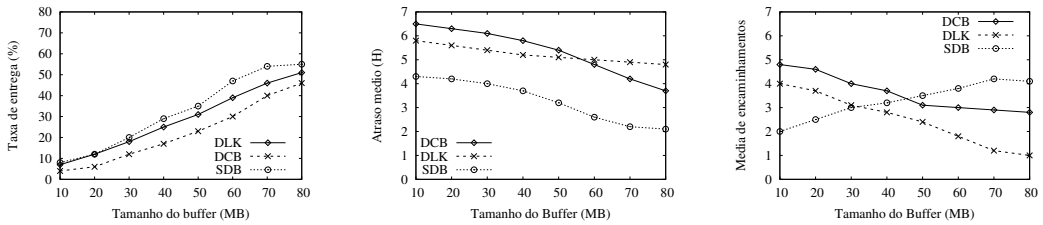


Figura 2. *Resultados para o trace Reality*

do aumento do espaço em *buffer*. Por outro lado, pode-se atribuir ao tempo entre contatos a razão para que o atraso médio do algoritmo SDB não sofra uma diminuição mais significativa mesmo com o aumento do espaço em *buffer*. O algoritmo SDB procura manter no *buffer* de um nó mensagens destinadas a outros nós com quem ele tem uma amizade forte, baseando-se na suposição de que esse nó é um dos melhores candidatos a entregar tais mensagens aos seus destinatários. No entanto, mesmo que aumente a probabilidade de entrega, essa decisão pode afetar o atraso médio de entrega caso o tempo entre contatos entre os nós seja alto. Nos resultados obtidos, pode-se concluir que o tempo entre contatos foi bastante parecido em qualquer tamanho de *buffer* testado, por isso o atraso médio para todos os tamanhos de *buffer* sofreu uma leve variação.

5.3. Média de saltos

A média de saltos pode ser definida como a quantidade média de encaminhamentos necessários para que uma mensagem alcance seu destinatário. As Figuras 1.(c) e 2.(c) apresentam os resultados obtidos para a métrica média de saltos nos cenários de mobilidade Cambridge e *Reality*. Com relação à métrica média de saltos, a hipótese levantada era a de que a utilização do egoísmo no gerenciamento do *buffer* contribuiria para a diminuição da média de saltos, porém, de acordo com os resultados obtidos, tal hipótese mostrou-se falsa.

Como pode-se observar, o algoritmo DLK obteve o melhor resultado com relação à métrica média de saltos para a maioria dos tamanhos de *buffer* testados em ambos os cenários. O algoritmo SDB obteve um resultado diferente dos demais algoritmos testados. Para os algoritmos DLK e DCB, quanto maior o espaço em *buffer*, menor a quantidade de saltos necessários para entregar a mensagem ao destinatário. O algoritmo SDB obteve um desempenho contrário a isto, ou seja, quanto maior o espaço em *buffer*, mais saltos foram necessários em média para que uma mensagem alcançasse seu destinatário. A justificativa é que com uma quantidade maior de espaço em *buffer*, mais mensagens podem ser armazenadas e encaminhadas em contatos posteriores e menos descartes são necessários. Dessa forma, mensagens que antes eram descartadas, agoram podem ser encaminhadas a outros nós que supostamente também sejam bons candidatos a entregar a mensagem ao destinatário. Apesar de essa prática aumentar a probabilidade de entrega, aumenta também a quantidade média de encaminhamentos necessários para que uma mensagem chegue ao seu destinatário.

6. Conclusões e Trabalhos Futuros

Neste trabalho, apresentou-se um algoritmo de gerência de *buffer* para redes DTN baseada no egoísmo dos nós. O objetivo principal deste algoritmo é tomar a decisão de que

mensagem descartar, em caso de *overflow*, baseando-se no tipo de egoísmo do nó, o qual pode ser classificado em individual, social ou não egoísta, e levando em consideração também a força da amizade entre os nós.

Para validar o algoritmo proposto, um conjunto de experimentos considerando métricas relacionadas à entrega de mensagens na rede foi realizado em dois cenários de mobilidade real bem conhecidos: Cambridge e *Reality*. Apesar de se levantarem algumas hipóteses negativas com relação ao uso do egoísmo no gerenciamento do *buffer* para DTNs, os resultados obtidos indicam que em ambos os cenários, o desempenho do algoritmo proposto mostrou-se muito promissor e inclusive superou outros algoritmos de gerenciamento existentes na literatura em termos da taxa de entrega e do atraso médio de entrega, além de obter um valor razoável de média de saltos.

Como trabalhos futuros, pretende-se avaliar de maneira mais extensa o desempenho do algoritmo SDB, comparando-o com outros algoritmos existentes na literatura para DTN. Pretende-se também avaliar o algoritmo SDB em conjunto com outros algoritmos de roteamento bem conhecidos para redes DTN, inclusive algoritmos de roteamento baseados em características sociais. Além disso, pretende-se inserir no algoritmo SDB uma forma de classificar o egoísmo dos nós de maneira dinâmica, coletando informações daquele nó e aplicando algoritmos de aprendizagem de máquina ou sistemas de reputação já existentes na literatura.

Referências

Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and Weiss, H. (2007). Delay-tolerant networking architecture. *RFC4838, April*.

de Oliveira, C. T., Moreira, M. D., Rubinstein, M. G., Costa, L. H. M., and Duarte, O. C. M. (2007). Redes tolerantes a atrasos e desconexões. *SBRC Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Domingos, P. and Pazzani, M. (1997). On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130.

Eagle, N. and Pentland, A. (2006). Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268.

Fall, K. (2003). A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34. ACM.

Gama, J. and teorema de Bayes, O. (2012). Aprendizagem bayesiana introdução. *Universidade do Porto*, pages 1–22.

Keränen, A., Ott, J., and Kärkkäinen, T. (2009). The ONE Simulator for DTN Protocol Evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA. ICST.

Krifa, A., Baraka, C., and Spyropoulos, T. (2008). Optimal buffer management policies for delay tolerant networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON'08. 5th Annual IEEE Communications Society Conference on*, pages 260–268. IEEE.

Lewis, D. D. (1998). Naive (bayes) at forty: The independence assumption in information retrieval. In *Machine learning: ECML-98*, pages 4–15. Springer.

Li, Q., Zhu, S., and Cao, G. (2010). Routing in socially selfish delay tolerant networks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE.

Lindgren, A. and Phanse, K. S. (2006). Evaluation of queueing policies and forwarding strategies for routing in intermittently connected networks. In *Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on*, pages 1–10. IEEE.

Naves, J. F., Moraes, I. M., and Albuquerque, C. V. (2012). Lps e lrf: Politicas de gerenciamento de buffer eficientes para redes tolerantes a atrasos e desconexões. *Simpósio Brasileiro de Redes de Computadores (SBRC?12)*, pages 293–305.

Pentland, A., Eagle, N., and Lazer, D. (2009). Inferring social network structure using mobile phone data. *Proceedings of the National Academy of Sciences (PNAS)*, 106(36):15274–15278.

Rashid, S., Ayub, Q., Zahid, M. S. M., and Abdullah, A. H. (2011). E-drop: An effective drop buffer management policy for dtn routing protocols. *International Journal*, 13(7):8–13.

Resta, G. and Santi, P. (2002). An analysis of the node spatial distribution of the random waypoint mobility model for ad hoc networks. In *Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 44–50. ACM.

Settawatcharawanit, T., Yamada, S., Haque, E., Rojviboonchai, K., et al. (2013). Message dropping policy in congested social delay tolerant networks. In *Computer Science and Software Engineering (JCSSE), 2013 10th International Joint Conference on*, pages 116–120. IEEE.

Silva, D. R., Costa, A., and Macedo, J. (2012). Energy impact analysis on dtn routing protocols. *ExtremeCom 2012*, pages 1–6.

Souza, C., Mota, E., Galvao, L., Manzoni, P., and Cano, J. C. (2014). Drop less known strategy for buffer management in dtn nodes. In *Proceedings of the Latin America Networking Conference on LANC 2014*, LANC '14, pages 6:1–6:7, New York, NY, USA. ACM.

Tang, L., Chai, Y., Li, Y., and Weng, B. (2012). Buffer management policies in opportunistic networks. *Journal of Computational Information Systems*, 8(12):5149–5159.

Vahdat, A., Becker, D., et al. (2000). Epidemic routing for partially connected ad hoc networks. Technical report, Technical Report CS-200006, Duke University.

Xia, F., Liu, L., Li, J., Ma, J., and Vasilakos, A. V. (2015). Socially aware networking: A survey. *IEEE Systems Journal*, 9(3):904–921.

Yoon, J., Liu, M., and Noble, B. (2003). Random waypoint considered harmful. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 1312–1321. IEEE.

Zhu, Y., Xu, B., Shi, X., and Wang, Y. (2013). A survey of social-based routing in delay tolerant networks: positive and negative social effects. *Communications Surveys & Tutorials, IEEE*, 15(1):387–401.

Um Mecanismo Eficiente de Controle de Congestionamento para Redes Tolerantes a Atrasos e Desconexões

Juliano F. Naves[1,2], **Igor M. Moraes**[1]

[1]Laboratório MídiaCom, PGC-TCC / Instituto de Computação
Universidade Federal Fluminense

[2]Núcleo Informatizado de Memória e Pesquisa do IFRO (NIMPI)
Instituto Federal de Rondônia, *campus* Vilhena

juliano.naves@ifro.edu.br, igor@ic.uff.br

Resumo. *Protocolos de roteamento para Redes Tolerantes a Atrasos e Desconexões frequentemente utilizam a replicação para aumentar a resiliência e a taxa de entrega de mensagens. No entanto, a replicação não controlada de mensagens pode levar ao congestionamento da rede, impactando negativamente o seu desempenho. Este artigo propõe um mecanismo híbrido de controle de congestionamento simples e eficiente que tem como objetivo evitar a replicação desnecessária de mensagens e atenuar o congestionamento na rede. O mecanismo proposto baseia-se no fato de que a remoção de mensagens dos buffers dos nós deve ser persistente, ou seja, um nó não deve tornar a receber mensagens que já foram removidas de seu buffer. A análise considera quatro registros reais de mobilidade e três protocolos de roteamento bem conhecidos na literatura. Os resultados mostram que o mecanismo proposto resulta em uma redução da sobrecarga de transmissão de até 98,5%, além de aumentar a taxa de entrega e reduzir o atraso de entrega de mensagens.*

Abstract. *Routing protocols for Delay and Disruption Tolerant Networks frequently use message replication in order to increase the network resilience and the message delivery rate. However, uncontrolled replication may lead to network congestion, negatively impacting network performance. This paper proposes a simple and efficient hybrid network congestion control mechanism that aims at avoiding unwanted message replication and to reduce network congestion. The proposal is based on the fact that the removal of messages from nodes buffers must be persistent, i.e. a node must not receive messages that have already been removed from its buffer again. The analysis consider four real mobility traces and three well-known routing protocols. Results show the proposed mechanism reduces transmission overhead in up to 98.5% in addition to increasing delivery rate and decrease delivery latency.*

1. Introdução

Na arquitetura TCP/IP, para que haja comunicação entre um par de nós, assume-se que sempre existe um caminho fim-a-fim entre a origem e o destino de uma mensagem. No entanto, tal suposição pode não ser apropriada para modelos existentes de redes sem-fio, que são caracterizados pela grande variação das condições do meio de

transmissão e pela mobilidade dos nós. Desconexões frequentes da rede são resultantes destas características, ou seja, um caminho fim-a-fim pode nem sempre estar disponível ou até mesmo não existir entre os nós que desejam se comunicar. Nestes cenários, a utilização da arquitetura TCP/IP é pouco eficiente e, por isso, faz-se necessário o desenvolvimento de uma nova arquitetura, específica para essas redes, que são chamadas de Redes Tolerantes a Atrasos e Desconexões (*Delay and Disruption Tolerant Networks* - DTN) [Oliveira et al. 2007]. As redes DTN, são baseadas no paradigma armazena-carrega-e-encaminha. Nesse paradigma os nós da rede são dotados de *buffers* e podem armazenar persistentemente uma mensagem, caso não haja um caminho fim-a-fim entre origem e destino, até que uma oportunidade de encaminhamento apropriada surja. Essas oportunidades de encaminhamento são chamadas de contatos.

Vários protocolos de roteamento, com diferentes características, foram propostos para operarem em redes DTN [Cao e Sun 2013]. É comum que estes protocolos façam uso da replicação das mensagens como forma de aumentar a probabilidade de entrega de mensagens. No entanto, tal medida pode esgotar rapidamente os recursos da rede, levando ao congestionamento da mesma [Naves et al. 2012a, Naves et al. 2012b]. Portanto, mecanismos de controle de congestionamento são importantes para garantir que estas redes atinjam o desempenho desejado. Estes mecanismos são projetados para evitar ou mitigar os efeitos negativos causados pelo congestionamento em DTNs.

As técnicas de controle de congestionamento em DTNs podem ser divididas em reativas, proativas e híbridas. As técnicas proativas utilizam medidas que visam prevenir o acontecimento do congestionamento. Por outro lado, as técnicas reativas usualmente aguardam o acontecimento do congestionamento para que alguma ação seja tomada. Por sua vez, técnicas de controle de congestionamento híbridas combinam técnicas reativas e proativas. Neste contexto, segundo Silva *et al.*, adotar uma abordagem proativa pode melhorar significativamente o desempenho da rede [Silva et al. 2016a].

O objetivo deste trabalho, portanto, é propor e avaliar um mecanismo híbrido de controle de congestionamento para DTNs. O mecanismo baseia-se na premissa de que a remoção de mensagens dos *buffers* dos nós deve ser persistente, isto é, um nó não deve tornar a receber mensagens que já foram removidas de seu *buffer*. Para tanto, na proposta deste trabalho, os nós da rede mantêm uma lista contendo os identificadores das mensagens que foram removidas dos seus *buffers*. Desta forma, durante um contato os nós não solicitam novamente mensagens que já foram removidas. Ademais, ressalta-se que o mecanismo proposto é independente do protocolo de roteamento. Para a avaliação, são utilizados quatro registros reais de mobilidade: Rollernet, Infocom05, DieselNet e Shopping. Além disso, três protocolos de roteamento bem conhecidos na literatura são utilizados, são eles: Epidêmico, MaxProp e Prophet. Os resultados demonstram que o mecanismo proposto é eficiente, reduzindo a sobrecarga de transmissão de mensagens em até 98,5%, além de aumentar a taxa de entrega e reduzir o atraso de entrega de mensagens.

O restante deste trabalho é organizado como a seguir. Na Seção 2, os trabalhos relacionados são descritos. O mecanismo de controle de congestionamento proposto neste trabalho é apresentado na Seção 3. A Seção 4 apresenta o ambiente de avaliação da proposta, assim como os registros de mobilidade reais utilizados nas simulações. Os resultados obtidos através das simulações são apresentados na Seção 5. Finalmente, as conclusões alcançadas no decorrer do presente trabalho, bem como possíveis extensões a

este trabalho e prováveis trabalhos futuros são apresentadas na Seção 6.

2. Trabalhos Relacionados

O controle de congestionamento é uma preocupação em DTNs desde a concepção desta arquitetura de rede. Recentemente, esforços foram empregados na análise, caracterização [Soelistijanto e Howarth 2014, Silva et al. 2015b] e também na avaliação [Silva et al. 2016a, Silva et al. 2015a] de mecanismos de controle de congestionamento propostos pela comunidade. Silva *et al.* classificam os mecanismos de controle de congestionamento em proativos, reativos e híbridos [Silva et al. 2015b]. Segundo os autores, mecanismos de controle de congestionamento proativos, também conhecidos como mecanismos de prevenção de congestionamento (*congestion avoidance*), agem de forma preventiva para tentar evitar que o congestionamento ocorra na rede. Por outro lado, os mecanismos de controle de congestionamento reativos esperam o acontecimento do congestionamento para entrarem em ação. Além disso, são mencionados mecanismos híbridos, que combinam as abordagens proativa e reativa.

Em trabalhos posteriores, Silva *et al.* fazem estudos comparativos de mecanismos de congestionamento em DTNs [Silva et al. 2016a, Silva et al. 2015a]. Segundo os autores, a adoção de uma abordagem proativa ou híbrida pode melhorar significativamente o desempenho da rede. Com relação aos mecanismos reativos, estes podem não alcançar um desempenho satisfatório devido aos grandes atrasos de comunicação que podem existir em DTNs. Como os mecanismos reativos esperam o acontecimento do congestionamento, estes podem sofrer um grande impacto do atraso de comunicação inerente às DTNs, o que atrasa as decisões tomadas por estes mecanismo e consequentemente agrava o congestionamento na rede.

É interessante notar que embora existam propostas restritas ao controle de congestionamento, protocolos de roteamento frequentemente especificam mecanismos de controle de congestionamento para sua operação. Este é o caso específico do protocolo MaxProp [Burgess et al. 2006], que especifica um mecanismo híbrido de controle de congestionamento. Neste protocolo, as mensagens possuem uma lista de saltos (*hop list*) na qual a mensagem armazena os nós pelos quais ela já passou. Desta forma, um nó A envia uma mensagem M_1 para um determinado nó B, se e somente se, B não está na lista de saltos de M_1. Além disso, o nó A envia a mensagem M_1 para B, se e somente se já não tiver enviado esta mensagem para B anteriormente. Este caso em específico difere do caso anterior, visto que a mensagem M_1 pode ter trafegado por diferentes caminhos. O mecanismo da lista de saltos foi implementado para a avaliação conduzida neste trabalho e este mecanismo será mencionado daqui em diante como *Hop List* (HL). Os outros dois mecanismos utilizados pelo protocolo MaxProp para lidar com o congestionamento da rede são o envio de reconhecimentos positivos [Burgess et al. 2007, Naves e Moraes 2014] e a política de gerenciamento de *buffer*.

A despeito das inúmeras pesquisas existentes, o congestionamento em DTNs continua sendo objetivo de trabalhos recentes. Este é o caso do trabalho de Silva *et al.*, que propõe um mecanismo de controle de congestionamento que tem como base a aprendizagem de máquina [Silva et al. 2016b]. Por sua vez, Liu *et al.* propõem um mecanismo de controle de congestionamento que considera as características sociais do nó, nomeado *Social Awareness based Congetion Control* (SACC) [Liu et al. 2016]. As características

sociais também são abordadas por Wei *et al.* em sua proposta denominada *Context-Aware Congestion Control Approach* (CACC) [Wei et al. 2014]. No entanto, o número de variáveis que representam o contexto da rede e que são utilizadas por estes mecanismo pode dificultar a implementação dos mesmos.

Sendo assim, este trabalho propõe um mecanismo de controle de congestionamento híbrido e fácil de ser implantado, visto que utiliza somente informações locais e não exige a troca de informações entre os nós da rede.

3. O Mecanismo Proposto

O objetivo de uma política de gerenciamento de *buffer* é maximizar o desempenho da rede, mantendo em *buffer* as mensagens com maior probabilidade de serem entregues ao destino. Desta forma, evitando causar impacto negativo pelo descarte inadequado de uma mensagem. Em casos específicos, como quando a mensagem já chegou ao destino, o descarte de uma mensagem pode ter impacto positivo no desempenho da rede, visto que impede que uma mensagem que já chegou ao destino continue a ser replicada, diminuindo a eficiência da rede. Em face do exposto, é razoável considerar que uma mensagem que foi descartada do *buffer* de um determinado nó utilizando uma política de gerenciamento de *buffer* que atenda ao objetivo exposto anteriormente possui baixa probabilidade de entrega. Isto porque a política visa manter as mensagens com maior probabilidade de entrega e portanto descartar as mensagens com menor probabilidade de entrega. Logo, é plausível assumir que o envio de uma determinada mensagem M_1 para um determinado nó A, após este nó já ter removido esta mensagem de seu *buffer* através de uma política de gerenciamento de *buffer*, é ineficiente. Em outras palavras, o descarte de uma mensagem deve ser permanente. Esta é a premissa utilizada pela proposta deste trabalho.

No mecanismo proposto, daqui em diante chamado de *Message List* (ML), os nós da rede mantêm uma lista contendo os identificadores de todas as mensagens que já descartaram. Desta forma, no início de um contato entre dois nós, durante a troca dos vetores de mensagens, ambos não solicitam o envio de mensagens que já receberam em algum momento e que foram descartadas de seus *buffers*. Ressalta-se que o mecanismo ML é um mecanismo de controle de congestionamento híbrido, visto que atua tanto reativamente quanto proativamente. A característica reativa advém do fato do mecanismo basear-se em uma política de gerenciamento de *buffer* para o seu funcionamento. Por outro lado, o mecanismo ML é também proativo, visto que impõe *a priori* um limite máximo de réplicas e de encaminhamentos na rede. Isto é, seja n o número de nós presentes na rede, uma mensagem só poderá ser replicada n vezes na rede. Com relação aos encaminhamentos, estas estão limitadas a $n - 1$, visto que um nó pode transmitir uma mensagem somente uma vez para todos os seus vizinhos.

Neste trabalho, assume-se que os nós da rede possuem um espaço reservado suficientemente grande para armazenar todos os identificadores das mensagens descartadas por eles, de modo que o mecanismo ML funcione plenamente. No entanto, a seguir é feita uma breve análise da sobrecarga de armazenamento resultado do mecanismo ML. As mensagens podem ser identificadas unicamente através do identificador da fonte da mensagem e da marca de tempo (*timestamp*) desta mensagem. A especificação do protocolo *Bundle* [Scott e Burleigh 2007] restringe o tamanho máximo dos identificadores a 2046 bytes, representados como *strings*. Com relação à marca de tempo, na especificação

do protocolo *Bundle* ela possui tamanho variável. No entanto, a utilização de 4 bytes é suficiente para representar cerca de 136 anos, como acontece com a marca de tempo padrão Unix. Isto impõe uma sobrecarga de 2.050 bytes, ou 16.400 bits, por mensagem. Uma forma de reduzir esta sobrecarga é utilizar uma função de dispersão. Uma alternativa é a utilização do *Message-Digest algorithm* 5 (MD5). Desta forma, aplicando-se a função MD5 ao par identificador da fonte e marca de tempo, a sobrecarga pode ser reduzida a 128 bits por mensagem.

A possibilidade de que os nós não tenham espaço suficiente para armazenar todos os identificadores das mensagens descartadas durante toda a operação da rede é admissível. Neste caso, quando a lista utilizada pelo mecanismo de controle de congestionamento proposto neste trabalho tiver tamanho limitado, ela pode ser gerenciada de modo similar ao modo em que o *buffer* dos nós é gerenciado, isto é, através das políticas de gerenciamento de *buffer*. Então, quando ocorrer a sobrecarga da lista em questão, isto é, quando um novo identificador de mensagem precisar ser armazenado e a lista não tiver espaço suficiente, uma política para o gerenciamento desta lista é acionada para escolher um identificador a ser descartado, de forma a liberar espaço para o recebimento do novo identificador. Entre as heurísticas que podem ser utilizadas para o gerenciamento da lista, cita-se a bem conhecida FIFO (*First In First Out*). Além disso, uma nova heurística que considere o identificador utilizado menos recentemente, similar ao bem conhecido algoritmo de gerenciamento de *cache Least Recently Used* (LRU) ou a política de gerenciamento de *buffer Least Recently Forwarded* [Naves et al. 2012a].

Para efeito de análise preliminar, simulações foram executadas com memória finita para armazenar os identificadores das mensagens. O espaço reservado na memória para o armazenamento dos identificadores foi variado entre 10 e 500 identificadores. Duas estratégias para substituição de identificadores na lista foram implementadas: FIFO e LRU. Os resultados indicam que a partir de 100 identificadores o desempenho torna-se equivalente a utilização de memória infinita. De qualquer forma, assumir um espaço reservado de memória infinito na rede serve de guia para comprovar que a premissa da proposta está correta: as mensagens devem ser permanentemente descartadas.

Finalmente, observa-se que como não utiliza informações específicas do protocolo de congestionamento, como por exemplo, a probabilidade de entrega calculada por alguns destes protocolos, o mecanismo proposto é independente de protocolo de roteamento. Portanto, é possível classificá-lo como de fácil implantação.

4. Cenários de Avaliação

4.1. Conjuntos de Dados e Roteamento

Três protocolos bem conhecidos da literatura foram utilizados na avaliação do presente estudo, são eles, Epidêmico, Prophet e MaxProp. Destaca-se que os três protocolos lidam com a replicação de mensagens de maneira distinta e portanto, atingem diferentes níveis de congestionamento. O protocolo Epidêmico não impõe qualquer controle na replicação de mensagens, sendo o mais suscetível ao congestionamento. O protocolo Prophet utiliza a probabilidade de entrega calculada pelo próprio protocolo para decidir se uma mensagem deve ser replicada para outro nó. Mais especificamente, uma mensagem M_1 é replicada pelo nó A para o nó B somente se a probabilidade de B enviar esta mensagem ao destino D for maior do que a probabilidade de A enviar esta mensagem para o

destino D. Logo, o protocolo Prophet é menos suscetível ao congestionamento do que o protocolo Epidêmico. Por último, o protocolo MaxProp utiliza mecanismos que impõem um limite estrito de réplicas e encaminhamentos, como detalhado na Seção 2, logo, entre os protocolos avaliados, é o protocolo menos suscetível ao congestionamento.

Quatro registros reais de mobilidade, são utilizados para avaliar o desempenho da proposta. O conjunto de registros Rollernet [Tournoux et al. 2009] é resultado de um experimento no qual foram distribuídos 62 iMotes para voluntários. Este conjunto tem duração aproximada de 3 horas. No conjunto de registros Infocom05 [Hui et al. 2005] 41 iMotes foram distribuídos entre os participantes da conferência IEEE Infocom do ano de 2005. Este conjunto tem duração de aproximadamente 3 dias. Por sua vez, o conjunto de registros DieselNet possui 31 nós e é resultado de um experimento realizado pela Universidade de Massachusetts, que implantou uma plataforma de testes para DTNs usando seus ônibus [Zhang et al. 2007, Burgess et al. 2006]. Por último, o conjunto Shopping [Galati et al. 2015] é resultado de um experimento no qual 25 dispositivos foram distribuídos em um *shopping*.

4.2. Ambiente de Simulação

A proposta deste trabalho foi implementada no simulador *Opportunistic Network Environment* (ONE). As simulações foram executadas para os quatro conjunto de dados distintos, com os três protocolos de roteamento escolhidos, resultando em 12 cenários diferentes. Para efeito de comparação, o mecanismo *Hop List*, presente no protocolo MaxProp, como descrito na Seção 2, foi implementado nos protocolos Epidêmico e Prophet. Adicionalmente, uma versão do protocolo MaxProp sem a utilização do mecanismo HL também foi implementada. Desta forma, pretende-se avaliar os efeitos destes diferentes mecanismos em três protocolos que lidam de maneira distinta com a replicação de mensagens.

Os parâmetros de configuração das simulações também são resumidos na Tabela 1. O padrão de tráfego de mensagens para cada conjunto de registros é o seguinte. Para o Rollernet, 500 mensagens são geradas durante as duas primeiras horas de um tempo total de aproximadamente 3 horas de simulação. Para os conjuntos de dados DieselNet, Infocom05 e Shopping, foram geradas 1.000 mensagens nas primeiras, respectivamente, 143, 52 e 125 horas, aproximadamente, de um tempo de simulação total de aproximadamente 187, 72 e 132 horas. O período de inatividade na geração de mensagens tem como objetivo diminuir a quantidade de mensagens que não chegam ao destino devido ao encerramento abrupto da simulação. O tamanho configurado para as mensagens é de 1,0 MB, visto que redes DTN trabalham com agregados que podem conter várias mensagens. Ainda sobre o padrão de tráfego, em todos os cenários os nós maliciosos não são fonte e nem destino de nenhuma mensagem. Para evitar efeitos negativos da má configuração do TTL, nenhuma mensagem expira durante todo o período de simulação. Para os cenários Dieselnet, Infocom05 e Shopping, o tamanho do *buffer* foi configurado para 20 M, enquanto que no cenário Rollernet, devido a seu tempo de duração mais curto e ao menor número de mensagens geradas, o tamanho do *buffer* foi configurado para 10 M.

5. Resultados

O mecanismo de controle de congestionamento proposto neste trabalho é comparado através de simulações com o mecanismo HL, implementado pelo protocolo de

Tabela 1. Parâmetros das simulações e características dos cenários.

Parâmetros/Conjuntos	DieselNet	Infocom05	Rollernet	Shopping
Dispositivo	802.11	iMote	iMote	*Bluetooth*
Duração ($\approx$)	7,8 dias	3 dias	3 horas	5,5 dias
Número de nós	31	41	62	25
Número de mensagens	1.000	1.000	500	1.000
Tamanho das mensagens (MB)	1	1	1	1
Taxa de Transmissão (Mbps)	1	1	1	1
TTL	∞	∞	∞	∞
Tamanho do *Buffer* (MB)	20	20	10	20

roteamento MaxProp, como observado na Seção 2. Além disso, o mecanismo é comparado com uma versão não modificada dos protocolos escolhidos. Para avaliar estes mecanismos três métricas de desempenho são utilizadas: a taxa de entrega de mensagens, a sobrecarga de transmissão de mensagens e o atraso de entrega de mensagens. Os resultados apresentados foram obtidos através da média de 10 rodadas de simulação distintas. Para todas as médias apresentadas nas figuras, calcula-se um intervalo de confiança para um nível de confiabilidade de 95%, representado por barras verticais.

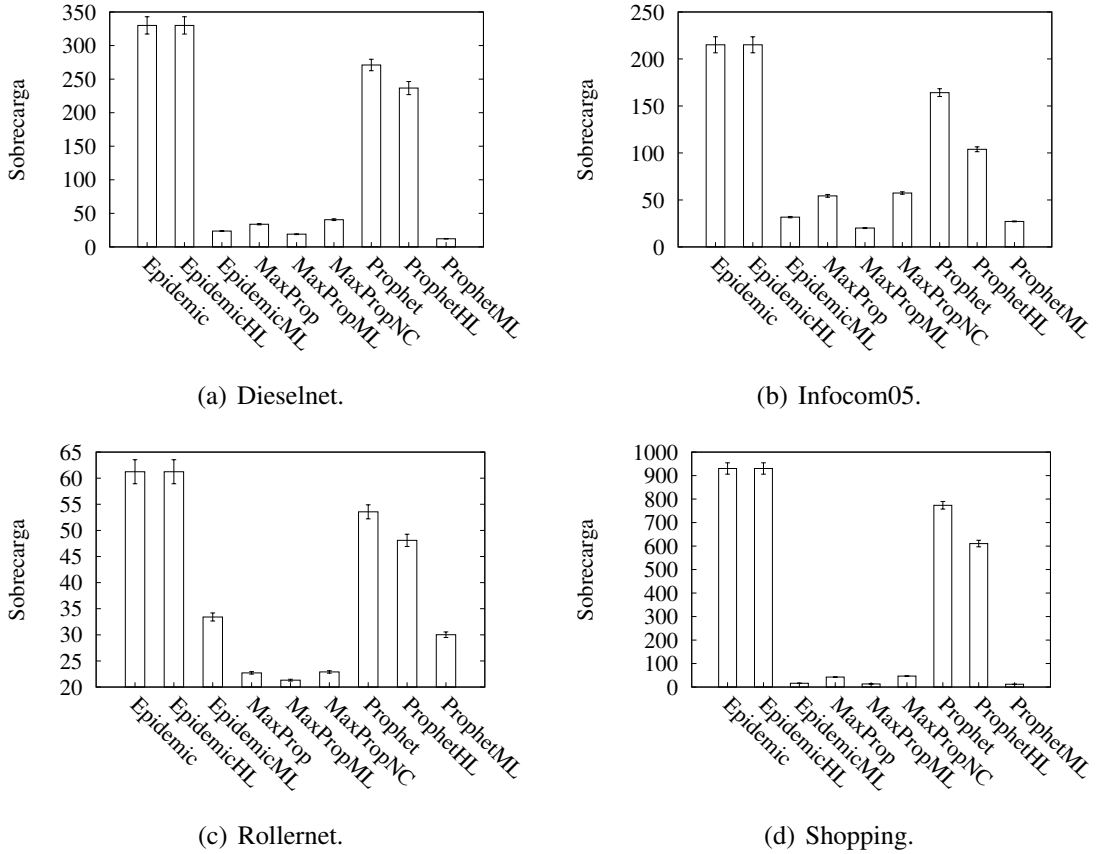
Nas figuras exibidas nesta seção, assume-se a seguinte nomenclatura. Epidemic, MaxProp e Prophet são os protocolos escolhidos para avaliação sem nenhuma modificação nos respectivos modos de operação. EpidemicML, MaxPropML e ProphetML são os protocolos dotados do mecanismo de controle de congestionamento proposto neste trabalho. EpidemicHL e ProphetHL são os protocolos Epidêmico e Prophet, respectivamente, dotados do mecanismo HL, que já é implementado no protocolo MaxProp por padrão. Por último, o nome MaxPropNC é atribuído a uma versão modificada do protocolo MaxProp que não implementa o mecanismo HL.

5.1. Sobrecarga

A sobrecarga S é calculada pela razão entre a diferença do número de mensagens encaminhadas m_t e mensagens entregues m_e e o número de mensagens entregues m_e, ou seja, $S = \frac{m_t - m_e}{m_e}$. Essa relação expressa a quantidade de mensagens adicionais que foram transmitidas para cada mensagem que chegou ao destino. Nos protocolos avaliados nesse trabalho, essa relação também indica a quantidade de réplicas criadas para cada mensagem entregue. Isto se deve ao fato destes protocolos replicarem as mensagens cada vez que a encaminham. Ou seja, mantêm a mensagem armazenada em *buffer*, mesmo após o encaminhamento. Adicionalmente, é interessante ressaltar que a sobrecarga de transmissão pode representar uma medida da eficiência energética dos protocolos, visto que a maior parte da energia de um nó sem-fio é consumida nos estados de transmissão e recepção de pacotes [Cunha et al. 2005].

A Figura 1 apresenta a sobrecarga de encaminhamentos para todos os cenários avaliados. Especificamente, a Figura 1(a) apresenta os resultados para o cenário Dieselnet, a Figura 1(b) apresenta os resultados para o cenário Infocom05, a Figura 1(c) apresenta os resultados para o cenário Rollernet e finalmente, a Figura 1(d) ilustra os resultados para o cenário Shopping.

É interessante observar que o mecanismo de controle de congestionamento pro-



(a) Dieselnet. (b) Infocom05.

(c) Rollernet. (d) Shopping.

Figura 1. Sobrecarga de encaminhamento para os cenários avaliados.

posto neste trabalho diminui a sobrecarga para todos os cenários avaliados. Para o cenário Dieselnet, como é possível observar na Figura 1(a), os protocolos ProphetML, MaxPropML e EpidemicML obtiveram a menor sobrecarga de transmissão. Observa-se que enquanto o protocolo Prophet original tem uma sobrecarga de cerca de 271 encaminhamentos, o protocolo ProphetML, que implementa a proposta deste trabalho, obtém uma sobrecarga de cerca de 12 encaminhamentos. Isto representa uma redução de cerca de 96% na sobrecarga. Por sua vez, o protocolo Epidêmico resultou em uma sobrecarga de 330 encaminhamentos enquanto que o protocolo EpidemicML resultou em uma sobrecarga de cerca de 24 mensagens, o que representa uma redução de cerca de 93% na sobrecarga de transmissão de mensagens. Por último, o protocolo MaxProp incorre em uma sobrecarga de transmissão de mensagens de cerca de 34 encaminhamentos, enquanto que o protocolo MaxPropML resulta em cerca de 19 encaminhamentos por mensagem entregue e isto uma redução de cerca de 44% na sobrecarga. É importante ressaltar que esta redução na sobrecarga de encaminhamento não tem impacto negativo na taxa de entrega. Pelo contrário, o mecanismo proposto leva a um aumento na taxa de entrega porque torna a rede mais eficiente ao evitar a replicação e o encaminhamento de mensagens com pouca probabilidade de entrega, como será discutido na Seção 5.2.

Passando para o cenário Infocom05, exibido na Figura 1(b), observa-se que os protocolos Epidêmico, Prophet e MaxProp resultam em uma sobrecarga de cerca de 215, 164 e 54 encaminhamentos, respectivamente. Por outro lado, as implementações que uti-

lizam a proposta deste trabalho, isto é, EpidemicML, ProphetML e MaxPropML, resultam em uma sobrecarga de cerca de 32, 27 e 20 encaminhamentos, respectivamente. Ou seja, a utilização do mecanismo ML resulta em uma redução de cerca e 85% da sobrecarga de transmissão para o protocolo Epidêmico, 83% para o protocolo Prophet e 63% para o protocolo MaxProp.

Seguindo com a avaliação do cenário Rollernet, ilustrado na Figura 1(c), é possível observar que este é o cenário com a menor sobrecarga de transmissão. Ainda assim, a utilização do mecanismo ML resultou em redução da sobrecarga de transmissão. Especificamente, a sobrecarga caiu de cerca de 61 encaminhamentos por mensagem entregue para o protocolo Epidêmico para cerca de 33 encaminhamentos para o protocolo EpidemicML, uma redução de cerca de 46%. Quando implementado no protocolo Prophet, o mecanismo proposto reduziu a sobrecarga de encaminhamento de cerca de 53 encaminhamentos para cerca de 30 encaminhamentos, uma redução de aproximadamente 43%. O protocolo MaxProp foi o protocolo que menos beneficiou-se da implementação da proposta no cenário Rollernet. Para este protocolo, a sobrecarga média caiu de cerca 22,7% para 22,89%, uma redução de menos de 1%.

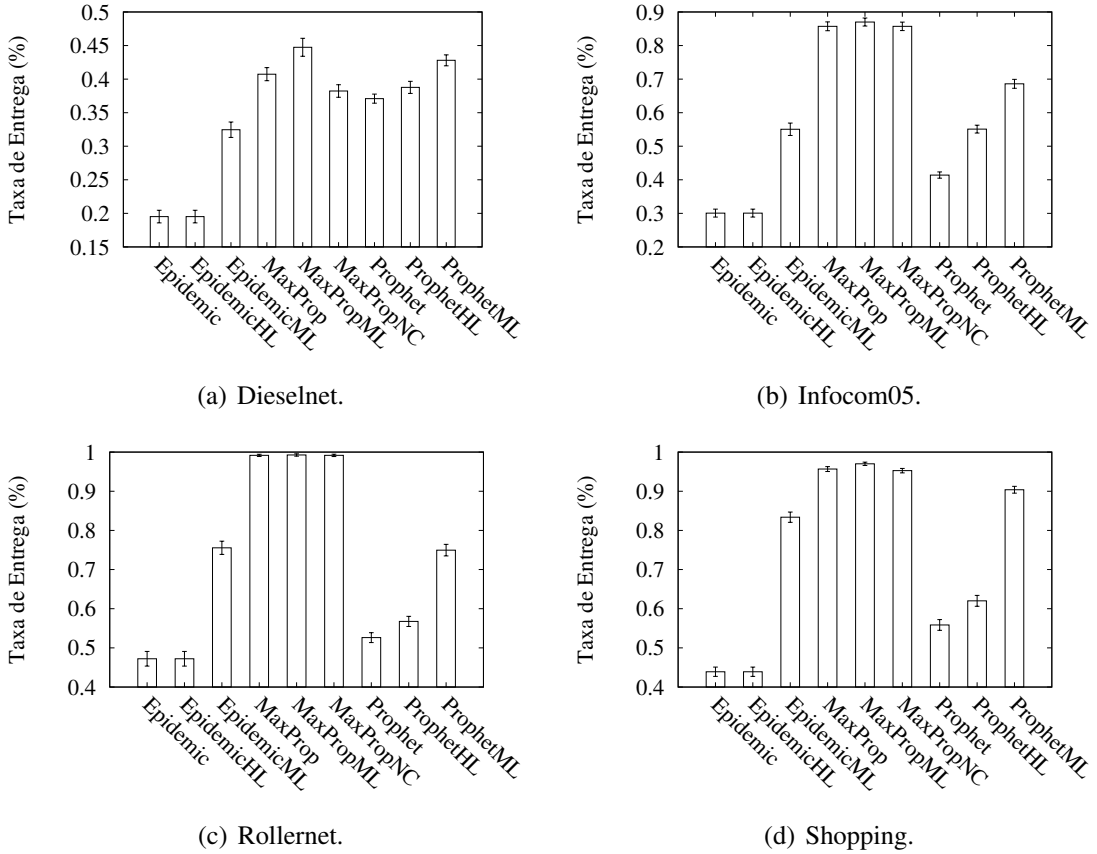
Por último, os resultados para o cenário Shopping são ilustrados na Figura 1(d). É possível observar que os protocolos Epidêmico, Prophet e MaxProp resultam, respectivamente, em uma sobrecarga de cerca de 930, 773 e 42 encaminhamentos. Por outro lado, quando utilizam o mecanismo de controle de congestionamento proposto neste trabalho, isto é, os protocolos EpidemicML, ProphetML e MaxPropML, resultam em uma sobrecarga de cerca de 16, 11 e 13 encaminhamentos, respectivamente. Ou seja, a utilização do mecanismo ML resulta em uma redução de cerca e 98% da sobrecarga de transmissão para o protocolo Epidêmico, 98,5% para o protocolo Prophet e 69% para o protocolo MaxProp.

Com relação ao mecanismo HL, embora a utilização deste mecanismo resulte em uma menor sobrecarga de encaminhamentos para os protocolos MaxProp e Prophet, nota-se que este mecanismo é superado pela proposta deste trabalho, que resulta em melhores resultados. Além disso, a proposta deste trabalho não incorre em sobrecarga de comunicação, diferentemente do mecanismo HL, visto que a lista de saltos é parte da mensagem e portanto, replicada junto com a mensagem.

5.2. Taxa de Entrega

A taxa de entrega é definida como o percentual de mensagens criadas que chegaram ao destino durante a simulação. Para as figuras aqui apresentadas, o eixo Y representa a taxa de entrega de mensagens obtida e os protocolos modificados e sem modificação são representados pelas barras. A Figura 2 apresenta as taxas de entrega para todos os cenários avaliados. Especificamente, a Figura 2(a) apresenta os resultados para o cenário Dieselnet, a Figura 2(b) apresenta os resultados para o cenário Infocom05, a Figura 2(c) apresenta os resultados para o cenário Rollernet e finalmente, a Figura 2(d) ilustra os resultados para o cenário Shoppingmall.

De modo geral, é possível observar que o mecanismo proposto, ML, aumenta a taxa de entrega alcançada para todos os protocolos avaliados. Isto ocorre porque o mecanismo proposto aumenta a eficiência da rede ao evitar o encaminhamento e a replicação de mensagens com baixa probabilidade de entrega. Além disso, é possível observar que



(a) Dieselnet. (b) Infocom05.

(c) Rollernet. (d) Shopping.

Figura 2. Taxa de entrega para os cenários avaliados.

o ganho de desempenho do mecanismo proposto é maior para os protocolos que atingem um nível de congestionamento maior, como é o caso dos protocolos Epidêmico e Prophet, assim como mencionado na Seção 4.1.

No cenário Dieselnet, a implementação do protocolo MaxProp que utiliza o mecanismo proposto para controle de congestionamento, denominada na figura como MaxPropML, obtém o melhor desempenho com relação a taxa de entrega entre todos os protocolos avaliados. Neste cenário, o MaxPropML alcançou uma taxa de entrega de cerca de 45%, seguido em desempenho pelo ProphetML, que atingiu cerca de 43% de taxa de entrega. Com relação ao protocolo MaxProp original, o protocolo MaxPropML obteve uma melhora de 5 pontos percentuais, ou neste caso, uma melhora de 12,5%.

Para o cenário Infocom05, o protocolo MaxPropML alcançou 87% de taxa de entrega, contra cerca de 86% do protocolo MaxProp original. Por sua vez, o protocolo ProphetML alcançou cerca de 69% contra cerca de 41% do protocolo Prophet original. Finalmente, o protocolo EpidemicML alcançou cerca de 55% de taxa de entrega contra uma taxa de entrega de 30% alcançada pelo protocolo original. Isto representa um acréscimo de 83% na taxa de entrega.

Por sua vez, o mecanismo HL mostra alguns resultados interessantes que serão comentados a seguir. A primeira observação a ser feita é que para o protocolo Epidêmico, o mecanismo HL não trouxe melhorias. Esta observação é válida para as demais

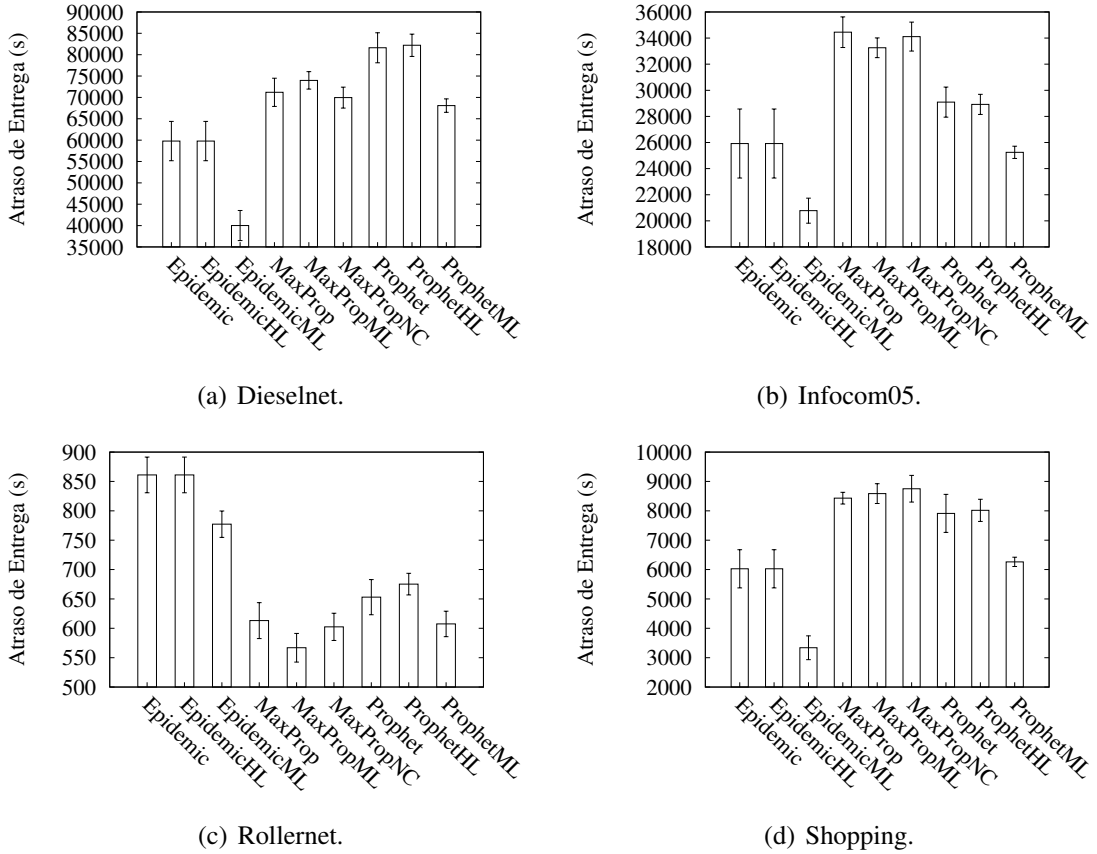
métricas de avaliação, abordadas nas Seções 5.3 e 5.1. Isto ocorre porque o protocolo Epidêmico não faz nenhum controle do número de réplicas ou mensagens encaminhadas na rede, desta forma, uma mensagem pode conter um número significativo de réplicas e cada uma destas réplicas pode ter trafegado por um caminho diferente. Sendo assim, é comum que mesmo que um determinado nó X já tenha recebido uma réplica da mensagem m_1 e posteriormente descartado esta réplica, novas réplicas de m_1 cheguem a X por caminhos distintos, que não passaram anteriormente por X e que por consequência, não se enquadram na premissa utilizada pelo mecanismo HL. Com relação ao protocolo Prophet, o mecanismo HL melhora o desempenho em todos os cenários avaliados. O caso do protocolo MaxProp é específico, pois este já implementa o mecanismo HL. Desta forma, para avaliar corretamente a melhoria de desempenho do mecanismo HL, decidiu-se implementar uma versão deste protocolo, chamada neste trabalho de MaxProNC. Desta forma, conseguiu-se observar que o mecanismo HL leva a ganhos de desempenho no protocolo MaxProp, principalmente em cenários menos conectados e por consequência mais congestionados, como exibido no cenário Dieselnet e apresentado na Figura 2(a).

5.3. Atraso de Entrega

O atraso de entrega ilustra a quantidade de tempo, em segundos, que decorre desde a criação de uma mensagem até o momento em que ela chega ao destinatário. Para os gráficos de atraso de entrega, apresentados na Figura 3, o eixo Y representa o atraso médio de entrega em segundos e os protocolos modificados e sem modificação são representados pelas barras. Especificamente, a Figura 3(a) apresenta os resultados para o cenário Dieselnet, a Figura 3(b) apresenta os resultados para o cenário Infocom05, a Figura 3(c) apresenta os resultados para o cenário Rollernet e finalmente, a Figura 3(d) ilustra os resultados para o cenário Shopping.

É possível observar que o mecanismo de controle de congestionamento proposto, ML, diminui o atraso de entrega na maioria dos cenários avaliados. Isto ocorre porque o mecanismo ML evita encaminhar e replicar mensagens com baixa probabilidade de entrega. Ao dar prioridade as mensagens com maior probabilidade de entrega, o mecanismo contribui para que estas mensagens sejam entregues mais rapidamente. As principais diferenças são apresentadas pelo protocolo Epidêmico. No cenário Dieselnet, apresentado na Figura 3(a), o atraso de entrega cai de cerca de 60.000 segundos, quando o protocolo original está sendo utilizado, para cerca de 40.000 segundos quando o mecanismo proposto é implementado no protocolo, uma redução de aproximadamente 5,5 horas ou cerca de 33%. Para o cenário Infocom05, apresentado na Figura 3(b), o atraso de entrega cai de cerca de 26.000 segundos para cerca de 21.000 segundos. Uma redução de cerca, de 1,4 horas, ou de cerca de 20%. Por sua vez, no cenário Rollernet, o atraso de entrega médio cai de cerca de 861 segundos para cerca de 777 segundos, quando o mecanismo de controle de congestionamento está sendo utilizado, o que corresponde a uma redução de cerca de 10% no atraso de entrega. Por último, no cenário Shopping, como exibido na Figura 3(d), o atraso de entrega médio obtido pelo protocolo Epidêmico cai de cerca de 6.026 segundos para 3.338 segundos, quando o mecanismo proposto é utilizado. Isto corresponde a uma redução de aproximadamente 45% do atraso de entrega médio.

Com relação ao protocolo Prophet, assim como no protocolo Epidêmico, a implementação da proposta reduziu o atraso de entrega em todos os 4 cenários avaliados. No entanto, algumas observações devem ser feitas a respeito do protocolo MaxProp. Neste



(a) Dieselnet. (b) Infocom05.

(c) Rollernet. (d) Shopping.

Figura 3. Atraso de entrega para os cenários avaliados.

protocolo, a utilização do mecanismo ML levou a um aumento no atraso de entrega em dois dos cenários avaliados, são eles, Dieselnet e Shopping, ilustrados nas Figuras 3(a) e 3(d), respectivamente. No entanto, os resultados são estatisticamente equivalentes, como é possível observar através do intervalo de confiança calculado.

Também é possível observar que o protocolo Epidêmico obteve o melhor desempenho com relação ao atraso de entrega para os cenários avaliados, exceto para o cenário Rollernet. Esta exceção ocorre devidos aos fatos que serão descritos a seguir. O conjunto de mobilidade Rollernet é notavelmente conectado, alterando períodos de alta conectividade com períodos menos conectados, fenômeno que os autores chamaram de "efeito acordeão" [Tournoux et al. 2009]. Esta alta conectividade ocasiona congestionamento, visto que possibilita que os nós repliquem exaustivamente as mensagens que possuem armazenadas em *buffer*. Observando-se que o protocolo Epidêmico não possui mecanismos para o controle de replicação, conclui-se que este cenário, combinando protocolo Epidêmico ao conjunto de mobilidade Rollernet, é o mais suscetível ao congestionamento dentre os cenários avaliados. Por outro lado, a despeito do aumento na taxa de entrega, o protocolo Epidêmico que implementa o mecanismo proposto atinge o menor atraso de entrega nos cenários Dieselnet, Infocom05 e Shopping. Isto demonstra que o mecanismo proposto oferece um equilíbrio entre o número de réplicas e a capacidade da rede, diminuindo o atraso de entrega enquanto aumenta ou mantém a taxa de entrega.

Por último, com relação ao mecanismo HL, é possível observar que este meca-

nismo levou a um pior desempenho com relação ao atraso de entrega para todos os cenários avaliados.

6. Conclusões e Trabalhos Futuros

Este trabalho propõe e avalia um novo mecanismo de controle de congestionamento para DTNs, nomeado ML. Esta proposta baseia-se na premissa de que o descarte das mensagens dos *buffers* dos nós deve ser permanente. Além disso, a proposta não incorre em sobrecarga de comunicação.

O mecanismo ML foi implementado em três protocolos bem conhecidos da literatura e seu desempenho foi comparado através de simulações com outro mecanismo utilizado para controle de congestionamento, o qual tem complexidade baixa, similar a proposta. Para as simulações, foram utilizados 4 conjuntos reais de mobilidade e para a avaliação três importantes métricas de desempenho foram utilizadas: taxa de entrega, atraso de entrega e sobrecarga. A avaliação demonstrou que o mecanismo é capaz de reduzir a sobrecarga de transmissão de mensagens em até 98,5%, com melhorias também na taxa de entrega de mensagens e no atraso de entrega de mensagens.

Para trabalhos futuros, pretende-se avaliar o mecanismo proposto com outros protocolos de roteamento, além de compará-lo a mecanismos de controle de congestionamento mais complexos existentes na literatura e que exigem um maior nível de conhecimento do contexto da rede. Além disso, pretende-se avaliar mais detalhadamente a premissa assumida por nossa proposta, ou seja, de que a remoção das mensagens dos *buffers* dos nós deve ser persistente. Adicionalmente, pretende-se avaliar a variação de outras variáveis da rede, como o tempo de vida das mensagens e o tamanho do *buffer* dos nós.

Agradecimentos

Esse trabalho é apoiado pela TBE/ANEEL, CAPES, CNPq, FAPERJ, CTIC, PROPESP/IFRO e Proppi/UFF. Os autores agradecem pelos dados obtidos do arquivo CRAWDAD de Dartmouth College.

Referências

Burgess, J., Bissias, G. D., Corner, M. D., e Levine, B. N. (2007). Surviving attacks on disruption-tolerant networks without authentication. Em *Mobihoc*, páginas 61–70.

Burgess, J., Gallagher, B., Jensen, D., e Levine, B. N. (2006). MaxProp: Routing for vehicle-based disruption-tolerant networks. Em *IEEE INFOCOM*, páginas 1–11.

Cao, Y. e Sun, Z. (2013). Routing in delay/disruption tolerant networks: A taxonomy, survey and challenges. *Communications Surveys & Tutorials, IEEE*, 15(2):654–677.

Cunha, D. d. O., Costa, L. H. M. K., e Duarte, O. C. M. B. (2005). *Analyzing the Energy Consumption of IEEE 802.11 Ad Hoc Networks*, páginas 473–484. Springer US.

Galati, A., Djemame, K., e Greenhalgh, C. (2015). Analysis of human mobility patterns for opportunistic forwarding in shopping mall environments. *Social Network Analysis and Mining*, 5(1):1–14.

Hui, P., Chaintreau, A., Scott, J., Gass, R., Crowcroft, J., e Diot, C. (2005). Pocket switched networks and human mobility in conference environments. Em *ACM SIGCOMM WDTN Workshop*, páginas 244–251.

Liu, Y., Wang, K., Guo, H., Lu, Q., e Sun, Y. (2016). Social-aware computing based congestion control in delay tolerant networks. *Mobile Networks and Applications*, páginas 1–12.

Naves, J. F. e Moraes, I. M. (2014). Uma avaliação do ataque de falsificação de reconhecimentos positivos em redes tolerantes a atrasos e desconexões. Em *Anais do 32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, páginas 105–118.

Naves, J. F., Moraes, I. M., e Albuquerque, C. (2012a). LPS and LRF: Efficient buffer management policies for delay and disruption tolerant networks. Em *IEEE 37th Conference on Local Computer Networks (LCN)*, páginas 368–375.

Naves, J. F., Moraes, I. M., e de Albuquerque, C. V. N. (2012b). LPS e LRF: Políticas de gerenciamento de buffer eficientes para redes tolerantes a atrasos e desconexões. Em *Anais do 30º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, páginas 233–246.

Oliveira, C. T., Moreira, M. D. D., Rubinstein, M. G., Costa, L. H. M. K., e B.Duarte, O. C. M. (2007). Redes tolerantes a atrasos e desconexões. Em *Minicursos do Simpósio Brasileiro de Redes de Computadores*.

Scott, K. e Burleigh, S. (2007). Bundle Protocol Specification. RFC 5050.

Silva, A. P., Burleigh, S., Hirata, C. M., e Obraczka, K. (2015a). DTN congestion control unplugged: A comprehensive performance study. Em *Proceedings of the 10th ACM MobiCom Workshop on Challenged Networks*, páginas 43–48.

Silva, A. P., Burleigh, S., Hirata, C. M., e Obraczka, K. (2015b). A survey on congestion control for delay and disruption tolerant networks. *Ad Hoc Networks*, 25, Part B(0):480 – 494.

Silva, A. P., Burleigh, S., Hirata, C. M., e Obraczka, K. (2016a). Congestion control in disruption-tolerant networks: A comparative study for interplanetary and terrestrial networking applications. *Ad Hoc Networks*, 44:1–18.

Silva, A. P., Obraczka, K., Burleigh, S., e Hirata, C. M. (2016b). Smart congestion control for delay- and disruption tolerant networks. Em *13th Annual IEEE International Conference on Sensing, Communication, and Networking*, páginas 1–9.

Soelistijanto, B. e Howarth, M. (2014). Transfer reliability and congestion control strategies in opportunistic networks: a survey. *IEEE Communications Surveys and Tutorials*, 16(1):538 – 555.

Tournoux, P.-U., Leguay, J., Benbadis, F., Conan, V., de Amorim, M. D., e Whitbeck, J. (2009). The accordion phenomenon: Analysis, characterization, and impact on DTN routing. Em *IEEE INFOCOM*, páginas 1116–1124.

Wei, K., Guo, S., e Xu, K. (2014). CACC: A context-aware congestion control approach in smartphone networks. *Communications Magazine, IEEE*, 52(6):42–48.

Zhang, X., Kurose, J., Levine, B. N., Towsley, D., e Zhang, H. (2007). Study of a bus-based disruption-tolerant network: Mobility modeling and impact on routing. Em *ACM MobiCom*, páginas 195–206.

Alocação dinâmica de largura de banda com predição dos próximos GRANT em redes Long-Reach PON

Madson R. Araujo[1], **Alex S. Santos** [1],**Tamise S. França**[1], **Stéfani S. Pires**[1], **Maycon L. M. Peixoto**[1], **Ricardo Rios**[1], **Gustavo B. Figueiredo**[1]

[1]Instituto de Matemática – Universidade Federal da Bahia (UFBA)
Av. Adhemar de Barros, s/n, Ondina – 40170-115 – Salvador – BA – Brasil

{madsonra,gustavo}@dcc.ufba.br,{santos.alex,ricardoar}@ufba.br

tamise-santos@hotmail.com,stefani.pires@ifba.edu.br,mayconleo@gmail.com

Abstract. *Passive Optical Networks (PON) technology was designed to work as a point-to-multipoint architecture and it has been widely adopted due to its simplicity, low cost and scalability. Typical PON is limited to a range of 20 km, whereas Long Reach Passive Optical Network (LRPON) can connect devices within a distance about 100 km. As the distance increases, it is important to better control the propagation delay of messages sent/received between ONU and OLT. However, there exists situations in which the time spent to control the delay can affect the system overall performance. Aiming at overcoming this drawback, we present a new prediction-based algorithm that models the time intervals used to transmit messages from ONUs to OLT, allowing to forecast when ONUs will start a new transmission and, as a consequence, reducing the number of control messages. Results emphasize the importance of the proposed algorithm and how it outperforms methods widely adopted in the literature.*

Resumo. *As Redes Óticas Passivas (Passive Optical Network – PON) são uma tecnologia promissora, simples, de baixo custo e escalável. Uma PON típica possui um alcance de 20 km, enquanto que PONs de longo alcance conectam dispositivos a uma distância de aproximadamente 100 km. À medida que a distância aumenta, torna-se necessário realizar um melhor controle do atraso de propagação de mensagens enviadas/recebidas entre ONU e OLT. Este trabalho apresenta um algoritmo de predição que modela intervalos de tempos utilizados para transmitir mensagens de ONUs para OLT, permitindo estimar quando ONUs iniciarão uma nova transmissão a fim de reduzir o número de mensagens de controle e, em última instância, o atraso de transmissão. Resultados destacam a importância do algoritmo proposto e como ele supera métodos largamente utilizados na literatura.*

1. Introdução

A crescente oferta de serviços "*triple-play*" (vídeo, áudio e dados) tem aumentado a necessidade por largura de banda nas redes de acesso, posicionando as Redes Ópticas Passivas (do inglês *Passive Optical Networks* – PON) [Lam 2011] como uma tecnologia de destaque. Além disso, operadores estão atualmente focados no desafio de oferecer serviços *quad play*, em que a mobilidade é adicionada aos serviços *triple play*. Neste sentido,

a rede de última milha, aquela do ponto de ligação entre os provedores de acesso e os clientes, precisa oferecer alta capacidade de transmissão, além de ser eficiente e dinâmica.

Devido à sua simplicidade, baixo custo e escalabilidade, as PONs são consideradas uma tecnologia promissora para suportar essa demanda de dados nas redes de acesso. A natureza passiva de seus componentes permite oferecer menor custo por unidade de banda e reduzir o CAPEX (do inglês *Capital Expediture*) e o OPEX (do inglês *Operational Expenditure*) dos operadores [Kazovsky et al. 2012, Lam 2011]. Porém, apesar de tais benefícios, as PONs tradicionais possuem alcance reduzido, sendo limitadas a uma distância aproximada de 10 a 20 Km, o que restringe sua abrangência e aplicabilidade em muitos cenários.

Para superar essa limitação de alcance das PONs tradicionais, foi proposta a PON de longo alcance (do inglês *Long Reach*-PON – LR-PON), a qual combina a rede metro e a rede de acesso em uma única rede, garantindo assim redução de custos, de componentes e de complexidade de desenvolvimento. Com isso, o alcance das PONs pode chegar a 100 Km [Usmani et al. 2014]. Esse aumento da distância máxima de operação amplia a quantidade de usuários atendidos na rede de acesso.

Entretanto, o aumento do alcance propiciado pelas LR-PONs resulta em um incremento no RTT (do inglês *Round Trip Time*) entre ONU e OLT, o que pode até inviabilizar a utilização das LR-PONs em muitos cenários. Uma das razões para aumento do atraso nas LR-PONs decorre do funcionamento dos algoritmos de DBA (do inglês *Dynamic Bandwidth Assignment*) utilizados na alocação dinâmica de banda [Novak et al. 2013]. Como será discutido na Seção 2, geralmente o processo de reserva de recursos incorre num atraso de, pelo menos, um RTT para que os dados da ONU sejam transmitidos. Consequentemente, quanto maior o tempo de propagação entre ONU e a OLT, maior o atraso sofrido pelas aplicações suportadas pela rede.

Para reduzir o tempo necessário para a transmissão, algumas pesquisas propõem o uso de técnicas de predição do comportamento das OLTs. Assim, se a OLT funciona em ciclos de processamento, pode-se inferir sobre a chegada de novas mensagens de requisição para antecipar o processamento dos pedidos e realizar a devolução das autorizações. Adicionalmente, a OLT pode usar predição para inferir o excesso de dados presentes no *buffer* de transmissão para oferecer uma mensagem de autorização que contemple chegadas de dados posteriores ao envio da requisição [Zhu and Ma 2008]. No entanto, apesar de tais predições serem capazes de causar uma redução no tempo de acesso de todas as ONUs, o impacto no tempo individual de cada ONU é limitado e o escopo da redução é limitado a um ciclo, não se estendendo a múltiplos ciclos futuros.

Este artigo apresenta um novo algoritmo de DBA que utiliza ferramentas de séries temporais para modelar o comportamento de ONUs e predizer suas futuras requisições. Em oposição aos trabalhos existentes na literatura, o algoritmo proposto realiza uma predição individualizada do comportamento futuro das ONUs e concede, através do envio antecipado de permissões de acesso, uma sequência de instantes de tempo para transmissão de dados em diversos ciclos futuros. Conforme apresentado nos resultados obtidos, ao realizar tais reservas antecipadas, o algoritmo proposto permite diminuir o número de RTTs, reduzindo, assim, o atraso em ciclos futuros.

As demais seções deste artigo são organizadas da seguinte forma. A Seção 2

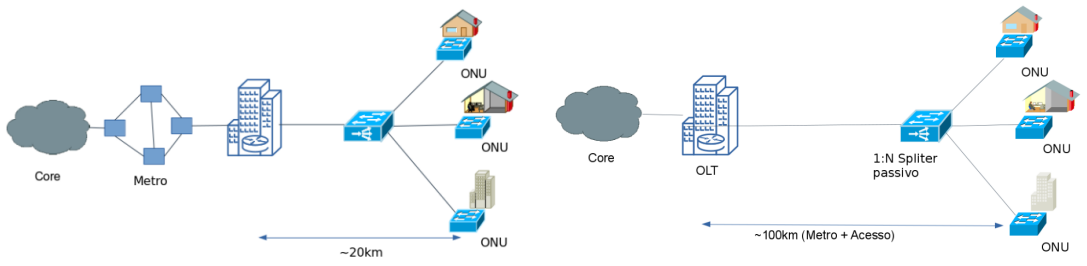
apresenta a tecnologia de redes óticas passivas. A Seção 3 descreve a alocação dinâmica de banda. A Seção 4 apresenta o algoritmo proposto. Na seção 5 são apresentados os resultados numéricos obtidos. Por fim, na seção 6 o trabalho é concluído.

2. Redes Ópticas Passivas

As redes ópticas passivas são caracterizadas por permitirem o compartilhamento de apenas uma fibra óptica entre os assinantes. Utilizam arquitetura de distribuição do tipo ponto-multiponto, podendo atingir distâncias entre 10 e 20 Km. Composta por, OLT (do inglês *Optical Line Terminator*), alocado no CO (do inglês *Central Office*), conectado às ONUs (do inglês *Optical Network Unit*), onde estão os usuários finais, usando um canal de comprimento de onda para *downstream* e outro para *upstream* passando por um único RN (do inglês *Remote Node*) passivo com funções de multiplexação [Sarkar et al. 2007, Harboe and Souza 2013, Lam 2011]. Ao chegar ao RN através de uma fibra de alimentação oriunda do OLT, o sinal óptico é dividido por um *splitter* passivo para fibras individuais, as fibras de distribuição que levarão o sinal até as ONU. A ausência de elementos ativos nesse trajeto, faz com que os equipamentos sejam menos suscetíveis a falhas e mais fáceis de manter [Feng and Ruan 2009, Yuksel et al. 2008].

2.1. *PON* DE LONGO ALCANCE

A *Long Reach PON* (LRPON) possui a mesma configuração básica de uma PON acrescida da rede metro, como mostra a Figura 1 [Song et al. 2010].



(a) PON tradicional (b) LR-PON

Figura 1. Arquitetura básica da LR-PON

A comunicação entre ONUs e OLTs em uma LR-PON se dá através de um determinado comprimento de onda. Na direção OLT-ONU, tem-se uma conexão do tipo ponto-multiponto e na direção oposta (ONU-OLT) tem-se a conexão multiponto-ponto. Quando a OLT deseja enviar dados para as ONUs, o envio é realizado em *broadcast*. Na direção oposta é necessário a presença de mecanismos que possam orquestrar o envio de informações de forma que não haja interferência entre ONUs [Song et al. 2010].

A principal particularidade das *Long Reach PONs* (LR-PONs) é o aumento da área de cobertura que pode passar dos 20km alcançados em PONs tradicionais chegando a aproximadamente 100km. Para isso, ela explora a utilização de amplificadores ópticos, dentre eles o *Erbium-doped fiber amplifier* (EDFA) e o *Semiconductor Optical Amplifier* (SOA), para compensar a perda de energia imposta pelo aumento da área de cobertura, diferentemente de PONs tradicionais que não possuem componentes ativos entre o CO e os usuários finais [Usmani et al. 2014, Song et al. 2010]. Além dos EDFAs e os SOAs,

há também os conversores opto-elétricos (OEO, do inglês *Optical Electrical Optical Converters*), que oferecem extensão de alcance através de regeneração de sinal. Uma desvantagem dos OEO é a sensibilidade à taxas de linha, e dessa forma, não podem permitir múltiplas taxas de linhas coexistentes [Skubic et al. 2010].

Uma LR-PON combina as redes metro e acesso para levar conexão de alta velocidade para as proximidades dos usuários [Song et al. 2010]. Em decorrência de tal junção, o CO passa a ter a função de conectar o núcleo da rede às redes de acesso. Além disso, ele se torna responsável por implementar as funções de *Layer* 2 e *Layer* 3, que são a alocação de recurso, agregação, gerenciamento e controle de serviço[Song et al. 2010].

Espera-se que LR-PONs passem a operar a uma taxa de 10 Gb/s e tenham de 2000 a 4000 ONUs [Kiaei et al. 2013]. Dentre as diversas topologias de rede propostas para LR-PON, podemos citar a *branch-and-tree* onde a fibra que compõe a seção de alimentação possui 90 Km (*tree*) e é dividida para múltiplos usuários (*branches*) no *local exchange* que reside na área do usuário. Há também a topologia *ring-and-spur*, onde a seção de alimentação é composta por um anel e os sinais ópticos são adicionados e removidos por *Optical Add-Drop Multiplexers* (OADMs) [Song et al. 2010].

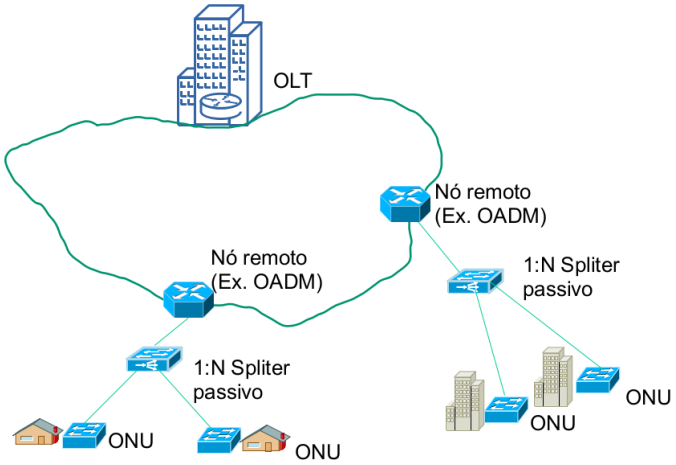


Figura 2. Topologia *ring-and-spur*. Adaptado de [Song et al. 2010]

Algumas implementações de LR-PONs são descritas na literatura. A *PLANET SuperPON* é um projeto desenvolvido pela *Advanced Communication Technologies and Services – Photonic Local Access NETwork* (ACTS-PLANET), que tem o objetivo de investigar possíveis *upgrades* de sistemas APON G.983 em termos de cobertura, fator de divisão, números de ONUs suportadas e taxas de transmissão. A arquitetura posposta por esse projeto que foi implementada no primeiro trimestre de 2000, suporta um total de 2048 ONUs e possui uma área de cobertura de 100 Km, onde os primeiros 90 Km correspondem à seção de alimentação e os 10 Km restantes à seção de distribuição [Song et al. 2010].

Uma outra implementação de redes LR-PON é a proposta pela *British Telecom*. Essa versão é equipada com 1024-*way split*, um alcance de 100 KM e taxa de transmissão de 10-Gbps em ambas as direções (*dowstream e upstream*). Seu fator de divisão é de 1024, fazendo com que menos amplificadores óticos sejam necessários se comparados com a *PLANET SuperPON* [Song et al. 2010]. O projeto desenvolvido pela *Photonic System Group* da *University College Cork, Irlanda* é composto de uma WDM-TDM LR-

PON híbrida que suporta múltiplos comprimentos de onda. Cada par de comprimento de onda pode suportar uma taxa de divisão de 256 usuários. Além disso, cada segmento de PON possui um alcance de 100 KM.

Tabela 1. Comparação entre implementações de LRPONs. Adaptado de [Song et al. 2010]

Implementação	Tecnologia	Alcance (km)	Comprimento de Onda	N. de ONUs
PLANET SuperPON	APON	100	1	2048
British Telecom	GPON	135	40	2560
Photonic System Group		100	17	4352

Com o aumento da área de cobertura nas LR-PONs, o tempo para troca de mensagens entre ONUs e OLTs sofre um atraso maior, que pode variar de 0.2 ms para 1 ms. Isso faz com que os mecanismos convencionais de alocação de banda se tornem ineficientes. Por isso, novos mecanismos para alocação dinâmica de banda devem ser abordados para superar tais limitações [Usmani et al. 2014].

3. Alocação dinâmica de banda

O uso de algoritmos de DBA proporciona maior eficiência do uso dos recursos por ofertar largura de banda conforme as demandas das ONUs [Hwang et al. 2008]. Eles utilizam o protocolo de controle multiponto, do inglês *Multipoint Control Protocol* (MPCP) para coordenar o acesso das ONUs à fibra compartilhada. Este protocolo define duas mensagens de controle que são usadas no processo de alocação de banda. A primeira delas é a mensagem *Report* e é usada pela ONU para solicitar alocação de banda. A segunda é a mensagem *Gate*, que é usada pela OLT para informar os detalhes da concessão (tempo de transmissão). Quando uma ONU necessita realizar o envio de dados para OLT, ela envia uma mensagem *Report* contento a quantidade em *bytes* que deseja transmitir, seu id e outras informações necessárias. A OLT envia a mensagem *Gate* para as ONUs informando a quantidade de bytes que ela pode transmitir, o id da ONU e demais informações necessárias [Song et al. 2009]. A Figura 3 ilustra o funcionamento do processo de reserva de recursos.

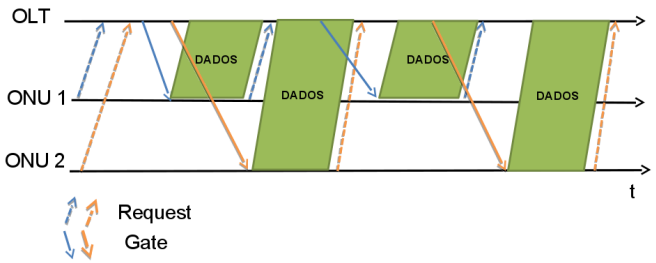


Figura 3. Troca de mensagens no processo de alocação de banda

3.1. Trabalhos relacionados

O IPACT [Kramer et al. 2002] é o algoritmo DBA das PONs tradicionais. Ele é um esquema de DBA centralizado para utilização da largura de banda de forma eficiente. Utiliza

duas mensagens de controle: *Grant* e *Report*. A principal vantagem deste algoritmo é que o tempo do ciclo se adapta de acordo com os dados disponíveis nos *buffers* dos ONUs. Contudo, o IPACT não é adequado para LR-PON [Novak et al. 2013]. Mesmo com baixa carga de dados o desempenho do IPACT é prejudicado devido ao aumento do RTT, que resulta em altas taxas de atraso.

No LSTP proposto em [Luo and Ansari 2005b], a solicitação dos recursos às próximas transmissões é realizada ao final do atual espaço de tempo em uso. A OLT ao receber a requisição com a previsão, aloca o menor valor entre a previsão contida na requisição e a largura máxima de banda determina pelo *Service Level Agreement* (SLA) de cada classe de tráfego. A ONU realiza a previsão das próximas demandas durante o período de espera baseando-se no tráfego que está sendo enfileirado e na classificação do tráfego. O tráfego é classificado com base no DiffServ [Luo and Ansari 2005a].

Em [Dixit et al. 2015], é proposto o *Synergized-Adaptive Multi-GATE polling with Void-filling* (S-AMGAV). Segundo o autor, a principal diferença de sua proposta para as demais existentes na literatura, é que a OLT emite mensagens GATE baseada na ocupação de cada ONU. Assim, a OLT não emite uma mensagem GATE para cada mensagem REPORT recebida.

Em [Zhu and Ma 2008] é proposto o algoritmo *Interleaved Polling With Adaptive Cycle Time with Grant Estimation* (IPACT-GE). Neste algoritmo, a ONU estima a quantidade de dados que chega entre dois *pollings* e com base nesta estimação a OLT decide o tamanho da transmissão concedida a ONU levando também em consideração a quantidade da requisição feita em ciclos anteriores. Para realizar a estimação, o método proposto se baseia nas características de alto-similaridade da rede. Em [Hwang et al. 2008], o mecanismo Early DBA realiza a predição conforme uma avaliação da variação histórica do tráfego requisitado por cada ONU. Com base na classificação do tráfego DiffServ.

Os autores em [Song et al. 2009] propõe o algoritmo *Multi-Thread Polling*. Nesta abordagem, as ONUs não utilizam o ciclo de requisição, confirmação e envio de dados. Uma determinada ONU pode realizar requisições seguidas antes de receber mensagens de confirmação da OLT criando assim a ideia de *thread*.

4. O Algoritmo Proposto

Esta seção apresenta o algoritmo *Predictive*-DBA (PD-DBA), que visa garantir alocação dinâmica de largura de banda e redução do atraso na rede com a diminuição de mensagens de requisição enviadas pelas ONUs. Inicialmente são apresentados os conceitos fundamentais sobre predição de dados utilizados como base para o desenvolvimento do algoritmo.

4.1. Predição de Dados com Série Temporal

O algoritmo de DBA proposto neste trabalho analisa o comportamento das fontes ONUs, modelando individualmente o padrão de suas requisições de envio de dados para as OLTs. Ao receber uma nova requisição para início de transmissão de dados, a OLT envia para a ONU solicitante o próximo instante livre e uma sequência de instantes futuros que podem ser utilizados sem a necessidade de realizar novas requisições. A sequência de instantes futuros é estimada por um preditor que modela informações históricas de início e fim de

transmissão de dados de cada ONU, possibilitando uma alocação dinâmica de largura de banda, conforme discutido a seguir.

Seja $\mathcal{D} = \{\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_n\}$ um ambiente de rede PON formado por n ONUs, sendo que cada ONU j pode ser definida como $\mathcal{O}_j = \{S_j, E_j\}$, tal que $1 \leq j \leq n$. No contexto deste trabalho, $S_j = \{s_0, s_1, \ldots, s_t\}$ e $E_j = \{e_0, e_1, \ldots, e_t\}$ representam duas séries temporais contendo, respectivamente, os instantes de início e fim de envio de dados da ONU j para a OLT. Por exemplo, a primeira transmissão da ONU j foi iniciada no instante s_0 e finalizada no instante e_0. A partir das informações contidas em S_j e E_j, uma função modelagem $f(\cdot)$ pode ser utilizada para predizer futuras requisições da ONU j.

Entretanto, devido ao grande volume de requisições realizadas no ambiente formado pelas ONUs e OLTs, a aplicação de técnicas tradicionais de modelagem de séries temporais possui duas grandes limitações. A primeira limitação ocorre porque é impraticável coletar todos os instantes de início e fim de transmissão de cada ONU para execução de um preditor em *batch*. Seria necessário, para isso, uma grande quantidade de memória e um alto custo de processamento. A segunda limitação deve-se à natureza das ONUs que requisitam, de maneira intermitente, novos instantes de transmissão, ou seja, é impossível armazenar previamente todas os instantes de tempo das requisições para realizar uma predição.

Para superar essas limitações, foram definidas duas variáveis: i) τ e ii) l. A primeira variável determina o tamanho de janela que será utilizada para modelar o comportamento dos instantes de início e fim de requisição de transmissão entre ONU e OLT, i.e., a predição utilizará um modelo obtido sobre as janelas $\{s_i, s_{i+1}, \ldots, s_{i+\tau}\}$ e $\{e_i, e_{i+1}, \ldots, e_{i+\tau}\}$, tal que $0 \leq i \leq t - \tau$. A segunda variável l define o número de observações preditas a partir do modelo obtido. Por exemplo, $f(S_j, \tau, l)$ é utilizada para predizer l instantes de início futuros com base em τ observações passadas.

Inicialmente, o preditor utilizado pelo algoritmo de DBA proposto aguarda até que τ requisições sejam realizadas. Durante esse período, cada solicitação de requisição de uma determinada ONU é respondida com um intervalo de início e fim de transmissão, que é determinado pelas políticas de escalonamento definidas nas OLTs. Após o preenchimento da janela, i.e., número de requisições por ONU for maior ou igual a τ, para cada nova requisição de transmissão, é enviado um intervalo definido pelo escalonador e uma sequencia l de predições, que podem ser utilizadas pela ONU solicitante em transmissões futuras sem a necessidade de realizar l novas solicitações para a OLT.

O tamanho τ da janela e o número l de predições podem ser ajustados em tempo de execução pela minimização do erro médio quadrático definido pela equação $e = E(\hat{\theta} - \theta)$. Nesta equação, $\hat{\theta}$ representa um conjunto de l instantes iniciais ou finais de transmissão preditos pelo algoritmo. Por outro lado, θ representa os instantes reais nos quais a ONU solicitante desejaria transmitir dados. Esses valores reais são recebidos pelo preditor na mesma mensagem enviada pela ONU para solicitar novos instantes de transmissão, após utilizar todos os instantes enviados na requisição anterior.

É importante destacar que o objetivo do algoritmo de DBA proposto não é encontrar a melhor modelagem e predição dos dados em redes Long-Reach PON, mas analisar como a predição pode reduzir a troca de mensagens entre ONUs e OLTs. Neste trabalho, a função $f(\cdot)$ de predição utilizada como prova de conceito realiza uma regressão linear

sobre os dados vez que a distribuição dos valores de início e fim de transmissão foram produzidos seguindo um modelo independente e identicamente distribuído com média e variância constantes [Hamilton 1994]. Melhores métodos de predição podem ser utilizados dependendo do padrão de trocas de mensagens no ambiente. Por exemplo, abordagens de modelagem de fluxos contínuos de dados para detecção de troca de contexto (*concept drift*) [Gama et al. 2014] podem melhorar a acurácia do preditor detectando, em tempo de execução, alterações de comportamento na transmissão de dados entre ONUs e OLTs.

4.2. Predictive-DBA

O algoritmo PD-DBA (Algoritmo 1) é invocado pelo OLT ao receber uma mensagem *report* de uma ONU_j. Ele recebe como entrada um *report* da ONU_j, o RTT do OLT até a ONU_j, um intervalo de guarda, e a largura de banda. Primeiro, o algoritmo verifica se há predições para a ONU_j. Em caso positivo, um conjunto de *grants* preditas é retornado à ONU_j. Caso contrário, o algoritmo calcula a janela de transmissão para a ONU_j e retorna uma única *grant* padrão contendo o inicio e fim da janela de transmissão da ONU_j.

Ao receber a *grant*, a ONU_j verifica se essa *grant* é padrão ou contém predições. Se a *grant* for padrão, o ONU_j envia os pacotes armazenados em seu *buffer* até o final da janela de transmissão concedida e em seguida envia um novo *report* informando o tamanho atual de seu *buffer*. Se a *grant* contiver predições, como mostrado na Figura 4, a ONU_j envia os pacotes armazenados no *buffer* até o final da primeira janela predita e em seguida espera até o inicio da próxima janela predita para transmitir novamente. Enquanto houver predições esse processo será repetido até a última janela predita. Ao final da transmissão da última janela predita, a ONU_j envia para OLT um *report* contendo o tamanho atual do seu *buffer* e um erro (período sem transmissão dentro da janela predita) associado a cada predição.

Entrada: ONU_j report buffer, ONU_j RTT, Intervalo de Guarda, Largura de Banda
Saída: Mensagem *Grant* especificando instantes de transmissão

```
1  início
2      predição,grant = getPredições(ONU_j);
3      if predição == True then
4          retorna grant;
5      end
6      else
7          tempo de envio = ONU_j report buffer/Largura de Banda ;
8          duração da grant = ONU_j RTT + tempo de envio + Intervalo de Guarda;
9          inicio da grant= tempo atual;
10         fim da grant= inicio da grant + duração da grant;
11         grant = [inicio da grant, fim da grant];
12         retorna grant;
13     end
14 fim
```

Algoritmo 1: PD-DBA

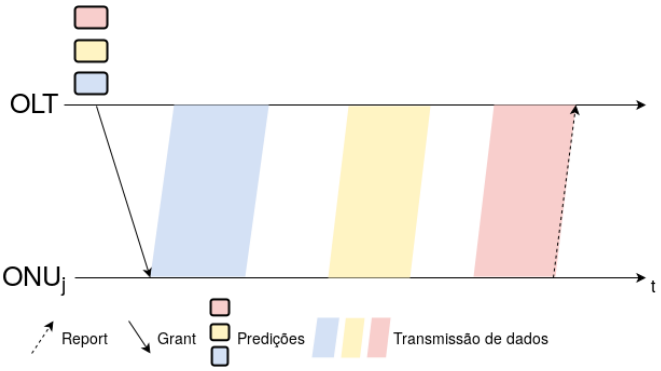


Figura 4. Exemplo do PD-DBA

5. Exemplos Numéricos

Para avaliar o desempenho do algoritmo proposto, foi desenvolvido um simulador *ad-hoc*, implementado em linguagem Python que utiliza a biblioteca Simpy[1]. A topologia utilizada na avaliação corresponde àquela apresentada na Figura **??**, na qual um conjunto de ONUs compartilha o mesmo canal *upstream* operando sob um mesmo comprimento de onda de 1Gb/s.

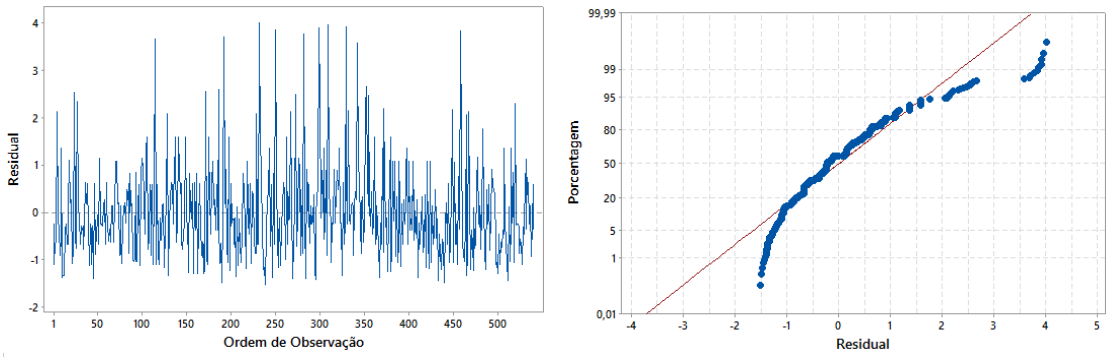
A carga de trabalho na simulação foi planejada usando pacotes (dentro da rede óptica) com tamanho fixo de 9KB (Jumbo Frame) seguindo a distribuição Poisson. Cada ONU possui *buffer* de tamanho ilimitado com *bucket* de 18KB e só pode enviar mensagem de *report* para OLT se a quantidade de dados no seu *buffer* for maior ou igual ao tamanho do *bucket*. O algoritmo proposto (PD-DBA) foi comparado com os algoritmos IPACT [Kramer et al. 2002] e OP-DBA (optimal dynamic bandwidth allocation), e os resultados foram obtidos operando sem limite de tamanho máximo de *grant*. O algoritmo OP-DBA utiliza como entrada um conjunto de *grants* alocados durante uma determinada execução IPACT e retorna esse conjunto de *grants* na forma de predições para as ONUs. Desse modo, o OP-DBA realiza uma predição perfeita das *grants* alocadas pelo IPACT. O objetivo do OP-DBA nesse trabalho é ser uma referência de predição que resulte em menor atraso, ou seja, uma referência representando o melhor caso para comparação com os demais algoritmos da mesma natureza.

Seguindo as orientações de [Jain 1991, Sanches and Wan 2015], os experimentos foram conduzidos utilizando o modelo de planejamento de experimentos fatorial completo. O projeto de experimentos desenvolvido neste trabalho considera os seguintes fatores que influenciam no desempenho do sistema: [Fator **A** - Algoritmo]= {IPACT, PD-DBA, OP-DBA}; [Fator **B** - Quantidade ONUs]={30, 60, 90}; [Fator **C** - Carga]= {37%, 74% 99%}; [Fator **D** - Distância]= {20km, 100km}. Os fatores e níveis do sistema foram definidos através de valores mais prováveis e utilizados na literatura, permitindo avaliar a completude do efeito que essas variáveis provocam ao ambiente. Todos os experimentos foram analisados a partir do intervalo de confiança[2], da média e do desvio padrão aferidos. Esses parâmetros são utilizados como base para o cálculo da soma dos quadrados, resultando na influência de cada fator nas variáveis de resposta.

[1]https://simpy.readthedocs.io/en/latest/index.html

[2]Utilização da distribuição *t-student* com replicações de 10 execuções por experimento e $\alpha = 0,05\%$.

Para validar os dados observados, a Figura 5(a) e a Figura 5(b) representam a ordem de execução dos experimentos e a distribuição residual observada nos resultados, respectivamente. Como pode ser observado na Figura 5(a), a qual apresenta os resíduos na ordem das observações correspondentes, a ordem das observações não influenciou os resultados, indicando que os experimentos não seguem uma ordem temporal. Este gráfico é útil, ainda, para demonstrar que há variação em relação às amostras, permitindo dizer que os resultados não são tendenciosos. Além disso, a Figura 5(b) apresenta a observação da normalidade na execução dos experimentos. O esperado é que os pontos do gráfico, relacionados aos experimentos, residam sobre ou próximos à linha normal, como é observado na Figura 5(b).

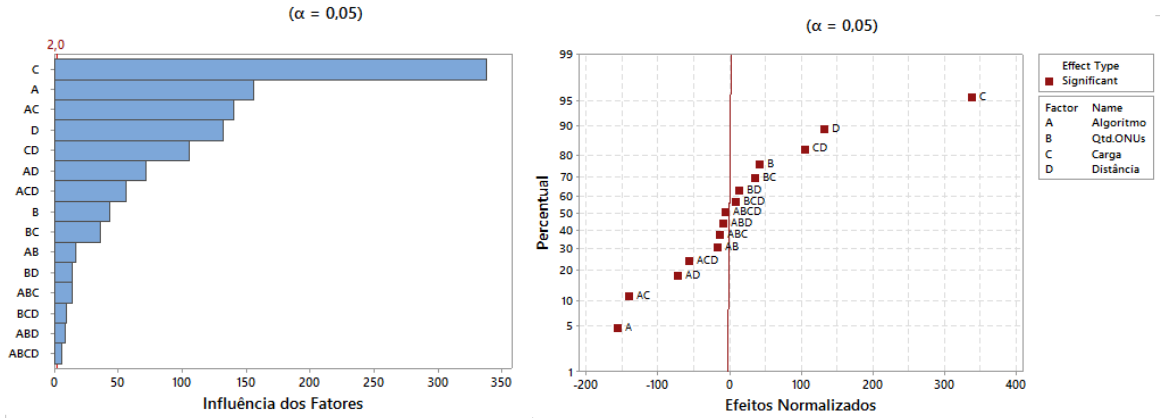


(a) Distribuição das Execuções

(b) Observação da Normalidade na Execução dos Experimentos

Figura 5. Validação das Observações dos Experimentos

A Figura 6(a) mostra o gráfico pareto dos efeitos para o projeto fatorial 2^k com combinação dos algoritmos OP-DBA e IPACT. Esse gráfico é utilizado para mostrar o grau de influência que os fatores exercem sobre a variável de resposta Atraso. Nota-se que o valor absoluto dos efeitos estão após a linha vermelha, indicando que todos os fatores escolhidos apresentam importância para o ambiente. Os efeitos mais significativos são a Carga (Fator C) seguido do Algoritmo (Fator A). Isso significa que mudanças nos níveis de carga e de algoritmo produzem uma alteração significativa na variável de resposta (Atraso).



(a) Influências dos Fatores

(b) Influência dos Fatores normalizada

Figura 6. Projeto Fatorial

Ainda lidando com a questão da influência dos fatores, a Figura 6(b) acrescenta mais informações com relação a esses efeitos observados. À medida que o fator C dirige-se à direita da linha vermelha normalizada, ocorre o acréscimo no valor obtido da variável de resposta. Por outro lado, à medida que o fator A se encontra à esquerda da linha normalizada, sugere-se uma diminuição do valor agregado obtido pela variável de resposta. Dessa forma, quando o cenário se encontra sobrecarregado, uma solução viável é tratar o Fator A (Algoritmo), pois esse fator é o responsável por produzir melhores resultados para a variável de resposta.

Dentro do conjunto de gráficos que a Figura 7 apresenta, existem informações com relação às interações entre os fatores dos experimentos. Apesar das retas estarem paralelas, não indicando interações significantes, nota-se que à medida que a carga agregada do ambiente (Qtd.ONUs, Carga e Distância) aumenta, existe uma inclinação das linhas em favor dos algoritmos PD-DBA e OP-DBA. Apesar de não possuir resultados ótimos como o algortimo OP-DBA, o algoritmo PD-DBA consegue melhorar seu desempenho quando os níveis de carga agregada aumentam.

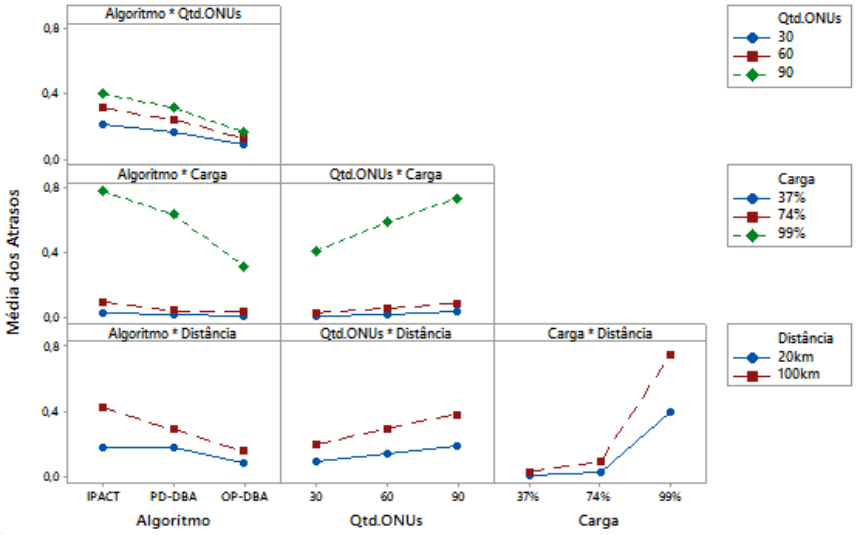


Figura 7. Interações entre os Fatores

A Figura 8 apresenta um gráfico análogo para ANOVA, testando a igualdade de médias da população por meio de uma aproximação normal com dados binomiais e de Poisson. Este gráfico é importante para testar a hipótese de que cada média da todas as combinações de fator/nível é igual à média geral de um nível especificado. Entretanto, o gráfico de Efeitos das Interações mostra que o algoritmo IPACT combinado com a distância 100 km rejeita a hipótese de que sua média é igual a média total, pois esse algoritmo utilizado para a rede de 100 km apresenta a valor de média superiores a média total do experimento. De fato, o gráfico de Principais Efeitos do Fator Algoritmo indica que o algoritmo IPACT supera o limite superior com referência a média total e o algoritmo OP-DBA supera o limite inferior com referência a mesma linha verde que possui a média de $0,2229$. O mesmo padrão se repete para a comparação das distâncias, mostrando que as médias das amostras estão fora dos limites de decisão na variação de 20 km para 100 km.

Um resumo dos efeitos dos fatores avaliados neste trabalho é apresentado na Fi-

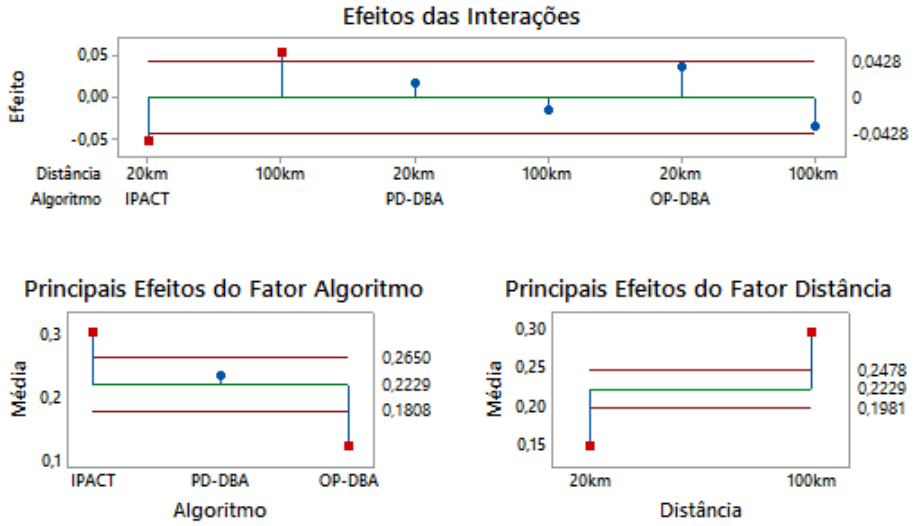
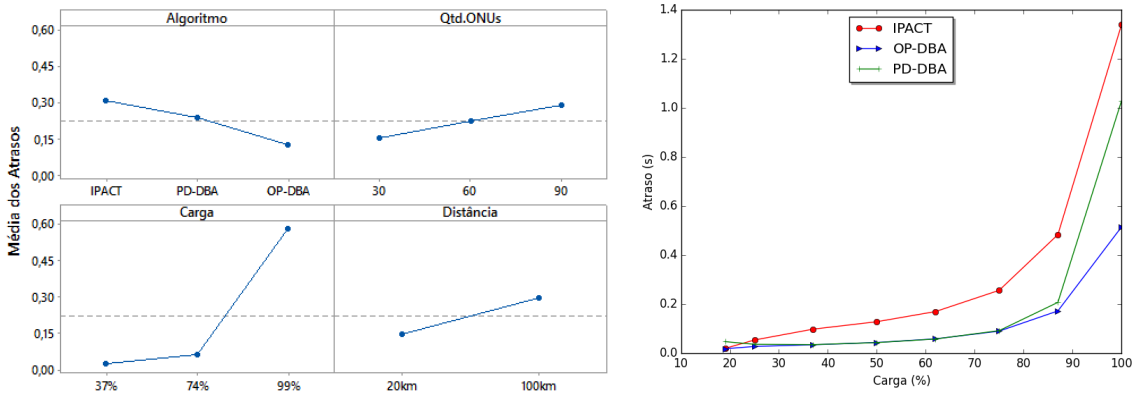


Figura 8. Two-Way (ANOVA)

gura 9(a). Em comparação com o IPACT, o algoritmo proposto neste trabalho (PD-DBA) conseguiu, por meio de suas predições, minimizar a média dos atrasos, mesmo em cenários de sobrecarga. Já a Figura 9(b) apresenta o comportamento dos algoritmos em função das diversas cargas impostas. De acordo com os cenários experimentados, o PD-DBA demonstrou ser capaz de gerenciar melhor as trocas de mensagens entre ONU e OLT na rede LR-PON do que o IPACT, causando menor atraso de propagação entre o tempo de viagem dos dados. Além disso, o PD-DBA apresentou atrasos médios próximos ao algoritmo de referência OP-DBA. Apesar do crescimento severo de tempo de atraso do PD-DBA na variação de taxa [87%-99%] de carga, ele permanece com tempos menores que o IPACT. Essa situação ocorre por causa da limitação do PD-DBA em gerar predições em função do aumento do número de requisições. Assim, nessa circunstância de tráfego elevado, o PD-DBA atua na forma convencional do IPACT.



(a) Principais Efeitos dos Fatores (b) Cenário com 90 ONUs da LR-PON

Figura 9. Comparação entre os principais efeitos

6. Conclusão

A LR-PON (*Long Reach* - PON) foi criada para superar as limitações de alcance das PONs tradicionais. Com o aumento do alcance da PON, ocorre também o aumento nos atrasos

de propagação das informações, surgindo a necessidade de otimizar o controle de troca de mensagem entre ONU e OLT. A hipótese adotada neste trabalho foi que a utilização de preditores baseados em séries temporais poderiam minimizar as trocas de mensagens entre a ONU e a OLT. Desse modo, foi proposto o algoritmo PD-DBA - (Predictive - DBA), um algoritmo baseado em predição que utiliza séries temporais para estimar a quantidade de dados que uma determinada ONU necessitará.

Simulações do algoritmo proposto (PD-DBA) mostraram que as predições podem minimizar a média dos atrasos, mesmo em função de cenários sobrecarregados, ou seja, mesmo quando a carga era de 99% com 90 ONUs solicitando requisições de GRANT simultaneamente. Os resultados também mostraram que o PD-DBA supera os valores conseguidos pelo IPACT em quase 26% na média, considerando todos os cenários dos experimentos utilizados. Além disso, o PD-DBA mantém-se próximo dos resultados obtidos pelo algoritmo de referência OP-DBA. Como trabalhos futuros, pretende-se refinar o algoritmo PD-DBA para aumentar a janela de predições, minimizando mais o atraso das requisições da ONU para novos GRANTs.

Agradecimentos

Os autores agradecem à CAPES, CNPq e FAPESB (Fundação de Amparo à Pesquisa do Estado da Bahia) pelo apoio recebido para o desenvolvimento deste trabalho.

Referências

Dixit, A., Lannoo, B., Colle, D., Pickavet, M., and Demeester, P. (2015). Synergized-adaptive multi-gate polling with void filling: Overcoming performance degradation in lr-pons. *IEEE/OSA Journal of Optical Communications and Networking*, 7(9):837–850.

Feng, T. and Ruan, L. (2009). Design of survivable hybrid wireless-optical broadband-access network. In *2009 IEEE International Conference on Communications*, pages 1–5.

Gama, J. a., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37.

Hamilton, J. D. (1994). *Time Series Analysis*. Princeton University Press.

Harboe, P. B. and Souza, J. R. (2013). Passive optical network: Characteristics, deployment, and perspectives. *IEEE Latin America Transactions*, 11(4):995–1000.

Hwang, I.-S., Shyu, Z.-D., Ke, L.-Y., and Chang, C.-C. (2008). A novel early dba mechanism with prediction-based fair excessive bandwidth allocation scheme in epon. *Computer Communications*, 31(9):1814–1823.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation, and modeling*. Wiley.

Kazovsky, L., Wong, S.-W., Ayhan, T., Albeyoglu, K. M., Ribeiro, M. R., and Shastri, A. (2012). Hybrid optical–wireless access networks. *Proceedings of the IEEE*, 100(5):1197–1225.

Kiaei, M., Fouli, K., Scheutzow, M., Maier, M., Reisslein, M., and Assi, C. (2013). Low-latency polling schemes for long-reach passive optical networks. *IEEE Transactions on Communications*, 61(7):2936–2945.

Kramer, G., Mukherjee, B., and Pesavento, G. (2002). Interleaved polling with adaptive cycle time (ipact): A dynamic bandwidth distribution scheme in an optical access network. *Photonic Network Communications*, 4(1):89–107.

Lam, C. F. (2011). *Passive optical networks: principles and practice*. Academic Press.

Luo, Y. and Ansari, N. (2005a). Dynamic upstream bandwidth allocation over ethernet pons. In *IEEE International Conference on Communications, 2005. ICC 2005. 2005*, volume 3, pages 1853–1857 Vol. 3.

Luo, Y. and Ansari, N. (2005b). Limited sharing with traffic prediction for dynamic bandwidth allocation and qos provisioning over ethernet passive optical networks. *Journal of Optical Networking*, 4(9):561–572.

Novak, T., Pilinsky, S. Z., and Draganić, M. (2013). Performance of dynamic bandwidth allocation algorithms in passive optical networks. In *Proceedings ELMAR-2013*, pages 157–160.

Sanches, M. and Wan, H. (2015). Work smarter, not harder: A tutorial on designing and conducting simulation experiments. *Winter Simulation Conference*.

Sarkar, S., Dixit, S., and Mukherjee, B. (2007). Hybrid wireless-optical broadband-access network (woban): A review of relevant challenges. *Journal of Lightwave Technology*, 25(11):3329–3340.

Skubic, B., Chen, J., Ahmed, J., Chen, B., Wosinska, L., and Mukherjee, B. (2010). Dynamic bandwidth allocation for long-reach pon: overcoming performance degradation. *IEEE Communications Magazine*, 48(11):100–108.

Song, H., Kim, B. W., and Mukherjee, B. (2009). Multi-thread polling: a dynamic bandwidth distribution scheme in long-reach pon. *IEEE Journal on Selected Areas in Communications*, 27(2):134–142.

Song, H., Kim, B. W., and Mukherjee, B. (2010). Long-reach optical access networks: A survey of research challenges, demonstrations, and bandwidth assignment mechanisms. *IEEE Communications Surveys Tutorials*, 12(1):112–123.

Usmani, F., Zaidi, S. M. H., Awais, A., and Raja, M. Y. A. (2014). Efficient dynamic bandwidth allocation schemes in long-reach passive optical networks- a survey. In *2014 11th Annual High Capacity Optical Networks and Emerging/Enabling Technologies (Photonics for Energy)*, pages 36–40.

Yuksel, K., Moeyaert, V., Wuilpart, M., and Mégret, P. (2008). Optical layer monitoring in passive optical networks (pons): A review. In *Proceedings of 10th Anniversary International Conference on Transparent Optical Networks (ICTON 2008), Athens, Greece*, pages 92–98.

Zhu, Y. and Ma, M. (2008). Ipact with grant estimation (ipact-ge) scheme for ethernet passive optical networks. *J. Lightwave Technol.*, 26(14):2055–2063.

Trilha Principal do SBRC 2017
Sessão Técnica 9
Engenharia de Tráfego e Balanceamento de Carga

A Control-based Load Balancing Algorithm with Flow Control for Dynamic and Heterogeneous Servers

Rodolpho G. de Siqueira[1], **Daniel R. Figueiredo**[1]

[1]Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro (UFRJ)
Caixa postal 68.511 – 21.941-972 – Rio de Janeiro – RJ – Brazil

Abstract. *Although load balancing is a fundamental and well-studied problem in resource allocation, the ever changing scenarios and technologies in distributed systems demand new approaches and algorithms. In this context, we consider a real world scenario where servers are heterogeneous and have dynamic background loads not controlled by the load balancer. In such scenarios, classic round robin policy or novel joint the shortest queue policy are not effective. We propose a load balancing algorithm that dispatches requests to a set of heterogeneous servers according to their CPU availability using a feedback control loop to prevent overloading. We implement this policy and evaluate its performance in real and controlled scenarios. Our evaluation indicates the proposed algorithm is more effective in distributing load than other classic policies, in particular when background load is dynamic.*

1. Introduction

Load balancing is a fundamental block in the design of many distributed systems since it provides for sharing demand (i.e., load) across a set of resources. Not surprisingly, load balancing mechanisms have been broadly studied from a theoretical perspective and is widely applied to real systems, most recently in the context of data centers and cloud services [Patel et al. 2013].

The tradeoffs and effectiveness of load balancing mechanisms greatly depend on the scenario under consideration. For example, load balancing web traffic in a data center is very different from load balancing backbone traffic in an ISP (Internet Service Provider). Underlying assumptions on the available information, such as the state of resources (e.g., current CPU load) and the demand (e.g., processing time of a request), as well as assumptions concerning dynamic aspects of resources and demand, greatly influence the design of an effective load balancing mechanism. Indeed, despite its long history, research and development on load balancing mechanisms continues [Gandhi et al. 2015], as different scenarios and assumptions emerge from technological advancements and new applications.

A common assumption in the design of load balancing mechanisms is that servers receive requests (i.e., load) only from the load balancer (one or more). Another common assumption is that the load balancer is rather "close" to the resources, in the sense that processing a request takes longer than sending the request to the resource. In this paper, we consider a real world scenario where these and other common assumptions do not hold, prompting the design of a new load balancing mechanism.

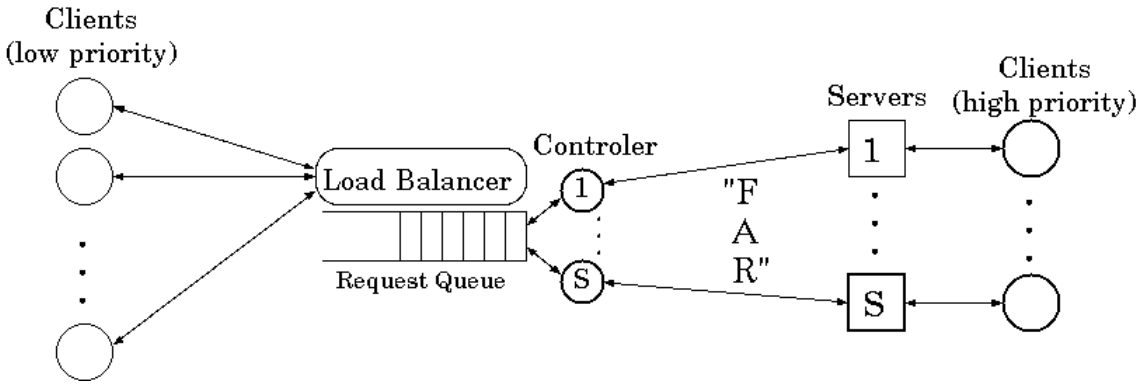


Figure 1. Illustration of the scenario considered where servers receive high priority demands from nearby clients while the load balancer dispatches low priority requests from far away.

In particular, we consider a scenario where a large set of clients need to monitor real-time variables stored on a small set of co-located servers. These real-time variables are replicated across the servers on local databases. However, the servers run mission critical applications that have high priority and unpredictable behavior. Thus, requests for reading real-time variables have low priority and must not impose a load that interfere with dynamic high priority demands. Last, servers have heterogeneous resources (e.g., different CPUs) and low priority clients are "far" from the set of servers (e.g, RTT is longer than the time to read a real-time variable). Figure 1 illustrates the scenario. This scenario is commonly found in oil platforms where inland clients need to monitor real-time variables in offshore oil platforms that run legacy servers and mission critical applications.

In this paper, we propose an effective load balancing mechanism for the above scenario. In particular, the load balancer dispatches aggregated requests to the servers according to a feedback control mechanism. The request rate sent to a server is determined by a corresponding feedback control mechanism that reacts to the CPU utilization on the server, an information periodically probed by the load balancer. We design the feedback control mechanism using a simple model for the relationship between number of requests and CPU load.

We implement the proposed mechanism and evaluate its performance under different scenarios, also comparing with the traditional Round Robing load balancing mechanism. Our results indicate that the proposed mechanism is effective in using the available resources in the servers without overshooting the pre-specified target CPU utilization. Moreover, the mechanism adapts to changes in CPU utilization of the servers due to high priority demands, moving low priority requests to under utilized servers.

The remainder of this paper is organized as follows. Section 2 poses the problem and the challenges in devising a load balancing mechanism in this context. Section 3 describes the feedback control for the proposed load balancing mechanism. Section 4 presents the evaluation of a real implementation of the proposed mechanism when running in different scenarios. Section 5 reviews the relevant work related to load balancing and control algorithms within the context of Dynamic Resource Provisioning. Finally, Section 6 exposes our conclusions and possible future work.

2. Problem formulation

As discussed in Section 1, we consider a real world scenario in which a small set of servers must respond with high priority to mission critical applications whose behavior are not easy to predict, but that remain idle for long periods of time. Thus, servers are mostly underutilized and their resources could be used more effectively, in particular when high priority applications are idle.

Consider the problem of remotely monitoring a very large set of real-time variables stored across this set of servers. Monitoring has low priority with respect to other applications running on these servers. Thus, resource usage by the monitoring application is restricted to a maximum budget, because servers must always be available to process high priority applications when needed. Moreover, since monitored variables are replicated across the servers, requests can be distributed across the servers, in particular proportionally to their CPU availability (and below the target) to minimize the chances of overloading the CPU when high priority applications suddenly run in any given server.

In summary, three basic goals must be achieved, which we discuss below:

1. Distribute low priority load to replicated but heterogeneous servers subject to uncontrolled high priority load.
2. Distribute low priority load to servers with CPU utilization below a pre-specified target.
3. Balance the low priority load across the servers in proportion to their CPU availability.

2.1. Load balancing

A common solution to load balancing is to adopt simple mechanisms such as Round Robin, and dispatch requests to servers accordingly. Nonetheless, since in our scenario the servers receive high priority demands from applications not subject to the load balancer, their load is not necessarily distributed proportionally to the available resources (i.e., CPU).

Other approaches use feedback signals from the servers to dispatch requests - for example, the balancer sends the request to the server with most available resources. When not even the most available server should receive further requests because it is operating with a CPU utilization above the pre-specified target, the balancer could simply refrain from sending requests. However, this kind of policy may lead the request rate to oscillate between two extremes, full-rate or no flow, which introduces jitter. As our client applications are for monitoring purposes, we propose a solution which leads to a steady, controlled flow instead. This results in less oscillations and monitoring samples taken at a more regular period.

2.2. Flow Control

A possible solution to the problem of avoiding overloading is to always dispatch low priority requests to servers and make them explicitly reject this requests when overloaded. However, such alternative can place an extra burden on an already loaded server, as rejecting small requests may cost almost the same as processing them. Also, this mechanism wastes a lot of time sending the request and waiting for the rejection, since in our scenario the servers are located far from the clients.

An alternative solution to avoid overloading the servers is to implement a throttling mechanism on the client side, making clients perceive timeouts as a sign of server overload and then slow down the request rate. However, this solution requires changing the code running on the client application. Thus, this alternative is not transparent to the clients and makes existing client implementations incompatible with the system (all clients must be changed).

Thus, we propose a joint solution which mixes load balancing with overload avoidance. Our solution meets the pre-specified target CPU utilization of the servers while simultaneously balancing the load proportionally to their CPU availability. Moreover, since the control intelligence is in the load balancer, our solution requires no change to the clients or the servers.

3. Proposed mechanism

We design a load balancer which does not simply distribute load equally among servers. Instead, it dispatches requests according to the servers' CPU availability, measured by an out-of-band feedback channel.

The balancer has a single request queue, from which requests are bundled and sent to the servers (see Figure 1). A maximum number of requests per second is removed from the queue and sent to a given server, as defined by the controller. The control is not performed jointly, but occurs independently for each server, following the same control algorithm. So, if two servers have the same amount of available resources, they will receive the same request rate; if one has more resources than the others, its controller will send it more requests; if a server's CPU utilization is above some target level, its request rate becomes negligible.

3.1. Controller

We propose a controller design which shall be applied independently to every server. Thus, we note that every parameter mentioned applies to a single controller instance, designed for a specific server.

In order to control the CPU utilization, we measure it at a constant frequency, taking samples every t_s seconds, which is the minimum time interval necessary for the measured value to change. If the controller takes decisions within a time interval smaller than t_s, then it will see the same previously measured value and may prematurely change its decision. Thus, we design a discrete-time controller which updates its control decision variable every t_s seconds, after having a new feedback measurement.

The balancer sends requests to a server as bundles. When it completes sending a bundle, it waits t_d seconds and then sends another one, which contains N_t requests to that given server. The value of N_t is decided for each server by the controller every t_s. Since the request rate to the servers at time instant t depends on the ratio between the number of requests N_t sent to the server and the interval t_d, we could control the rate by varying either of those values. As the timing requirements of the monitoring application for which the controller is intended are not very strict, we make t_d constant and vary N to achieve the request rate control.

In order to design the controller, we first determine a simple model of the steady-state relationship between CPU utilization of an idle server and the number of request

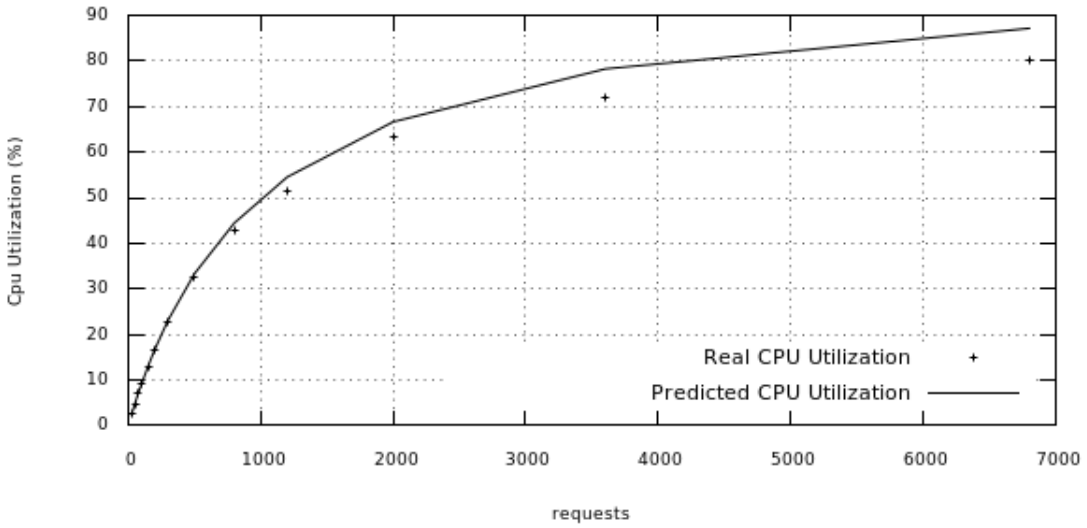


Figure 2. Comparison between model prediction and real system behavior (CPU utilization as a function of number of requests sent periodically).

received and processed every t_d seconds. This simple model assumes the CPU is idle - which is the case most of the time within our assumptions, as the high priority applications only runs sporadically.

Using this assumption, we estimate the CPU utilization through the ratio between the time spent processing a bundle of requests and the total time between the receiving of two consecutive bundle of requests. Let C_t denote CPU utilization at time t, N_t the number of requests in a bundle at time t, t_p the time required to process a single request, and t_d the period of the balancer (which sends a bundle of requests every t_d time units). Thus, we have the following ratio:

$$C_t = \frac{N_t t_p}{t_d + N_t t_p}, \tag{1}$$

where we have assumed that the network delay is negligible when compared to t_d and to the time required to process the bundle of requests, $N_t t_p$.

Note that this is a steady-state model while in practice there is a one-minute moving average dynamic relationship between the number of requests and the measured CPU utilization. We consider this dynamic relationship when operating the controller. Note also that t_p may be different across a set of heterogeneous servers, and hence the model can be adjusted for each case. Figure 2 shows the predicted value and the actual measured value of CPU utilization (in steady-state) as a function of the number of requests sent to the server, indicating a very good agreement, in particular when the number of requests is small. As the number of requests increases, the model assumption starts to be violated as the time required to transmit the requests become less negligible. Nonetheless, the model prediction is still in good agreement, and this simplified model has proved to be a good enough approximation for our purposes.

With the model relating C_t to N_t, we can evaluate how much N should vary in order for C to vary a desired amount. In particular, assuming a first-order approximation

of C as a function of N, we have the following:

$$\begin{aligned} C_t - C_{t-1} &= (N_t - N_{t-1})\frac{dC_{t-1}}{dN_{t-1}} \\ &= (N_t - N_{t-1})\frac{d}{dN_{t-1}}\left(\frac{t_p N_{t-1}}{t_d + t_p N_{t-1}}\right) \\ &= (N_t - N_{t-1})\frac{t_p t_d}{(t_d + t_p N_{t-1})^2} \end{aligned} \tag{2}$$

Considering that C is measured by a moving average, we cannot expect to control C in less time than the window over which the average is calculated, t_w. Hence, the controller should try to make C_t approach C^*, the CPU utilization set-point, by varying N through a process which takes place in an interval proportional to the time window t_w - the controller should take longer to achieve the reference if t_w is long, and make smaller variations to N. Note also that, since the controller only updates its control decision when it receives a feedback sample, once every t_s seconds, the variation in the number of requests should be proportional to t_s - if samples are not taken frequently, the measured CPU utilization shall have varied more from one sample to another, and then the controller needs to be more intense each time it has a chance to act.

With that reasoning, we come to the conclusion that C_t should approach C^* by a fraction $\frac{t_s}{t_w}$ of the difference $C^* - C_{t-1}$ every t_s seconds:

$$C_t - C_{t-1} = \frac{t_s}{t_w}(C^* - C_{t-1}) = (N_t - N_{t-1})\frac{t_p t_d}{(t_d + t_p N_{t-1})^2} \tag{3}$$

This yields a non-linear integral controller, which, at each instant t, increments the allowed request rate to a server based on a fraction of the difference between the reference CPU utilization, C^*, and its current measured value:

$$N_t = N_{t-1} + \frac{t_s}{t_w}\frac{(t_d + t_p N_{t-1})^2}{t_p t_d}(C^* - C_{t-1}) \tag{4}$$

4. Evaluation

We implemented the proposed load balancing mechanism and evaluated its performance in a controlled environment consisting of four Linux machines: a client, the load balancer, and two servers. We report on actual resource usage as reported by Zabbix, a common Network Monitoring tool. In particular, the load balancer machine runs a Zabbix poller, and the other machines run Zabbix agents that report the values for CPU utilization and network bandwidth.

We also evaluate and compare with the performance of a slightly modified round robin load balancer. This modifications sets a limit to the maximum request rate sent to each server in order to respect the target CPU utilization - using equation 1. However, this limit is computed assuming the servers are idle and is fixed - no adjustments to the maximum rate is performed during the experiments. Note that this modification avoids overloading any individual server when idle, allowing for a more direct comparison with our proposal.

We consider the following parameters: Target CPU utilization for the low priority requests of 15% (C^* in the model), time window over which the CPU utilization is measured is 1 minute (t_w in the model), and sampling period is 5 seconds (t_s in the model). Thus, the proposed controller repeatedly calculates, based on feedback values received every 5 seconds, the maximum request rate that can be allowed so that the CPU utilization does not violate the target.

Two different scenarios are considered for the evaluation and comparison of the load balancing mechanisms: infinite load and limited load for low priority requests. In the infinite load scenario, the client makes low priority requests as fast as possible: a single message with a large number of requests is sent to the load balancer and, as soon as the response message is received, another message with many requests is sent. This scenario evaluates the controller response more directly, since there is always enough demand to achieve different values of CPU utilization. In the limited load scenario, the client periodically sends a message with a limited number of requests in a rate that is not enough to reach the target CPU utilization when the servers are idle.

4.1. Infinite load scenario

In this scenario we start at time $t = 0$ with the two servers idle (no high priority traffic), and thus receiving and processing low priority requests. At time $t = 4$ minutes, a high priority demand arrives to Server 0, generating a CPU utilization of 25%, which lasts until time $t = 9$ minutes. Server 1 never receives any high priority demands.

Figure 3 shows the behavior of the two servers under the modified Round Robin policy. Note that the computed maximum request rate keeps the CPU utilization of the servers on target when servers are idle (until $t = 4$ minutes), balancing the requests equally among the two servers (note that incoming traffic to each server is 25Kbps). However, there is no reaction to the high priority demand, at time $t = 4$ to Server 0, and the load balancer maintains the request rate to both servers. Thus, it continues to impose a CPU load on Server 0 with low priority requests while high priority demand is processing and generating CPU utilization above the target. When high priority load finishes at time $t = 7$ minutes, the servers return to the target CPU utilization.

Figure 4 shows the behavior of the same scenario with the proposed load balancing controller. Note that until time $t = 4$ minutes the behavior is very similar to round robin policy, with the controller meeting the target CPU utilization and equally dividing the requests among the two servers (note incoming traffic to each server). However, when high priority demands arrive to Server 0, the controller reacts by reducing the low priority traffic, allowing the CPU to essentially exclusively handle the high priority demand, which demands 25% CPU utilization. Note that low priority traffic to Server 0 reduced from 25Kbps to 5Kbps. When high demand is over, at around time $t = 8$ minutes, the controller ramps up the low priority request rate to meet the target CPU utilization, at 15%.

As expected, the controller does not increase the request rate of Server 1, in order to compensate the the low request rate sent to Server 0. Note that Server 1 is already at its target CPU utilization, and can thus not handle a higher request rate. As a consequence, the client must wait more, as requests previously handled by Server 0 are not being dispatched. Indeed, Figure 5 shows the network traffic leaving the client. At time $t = 4$ we

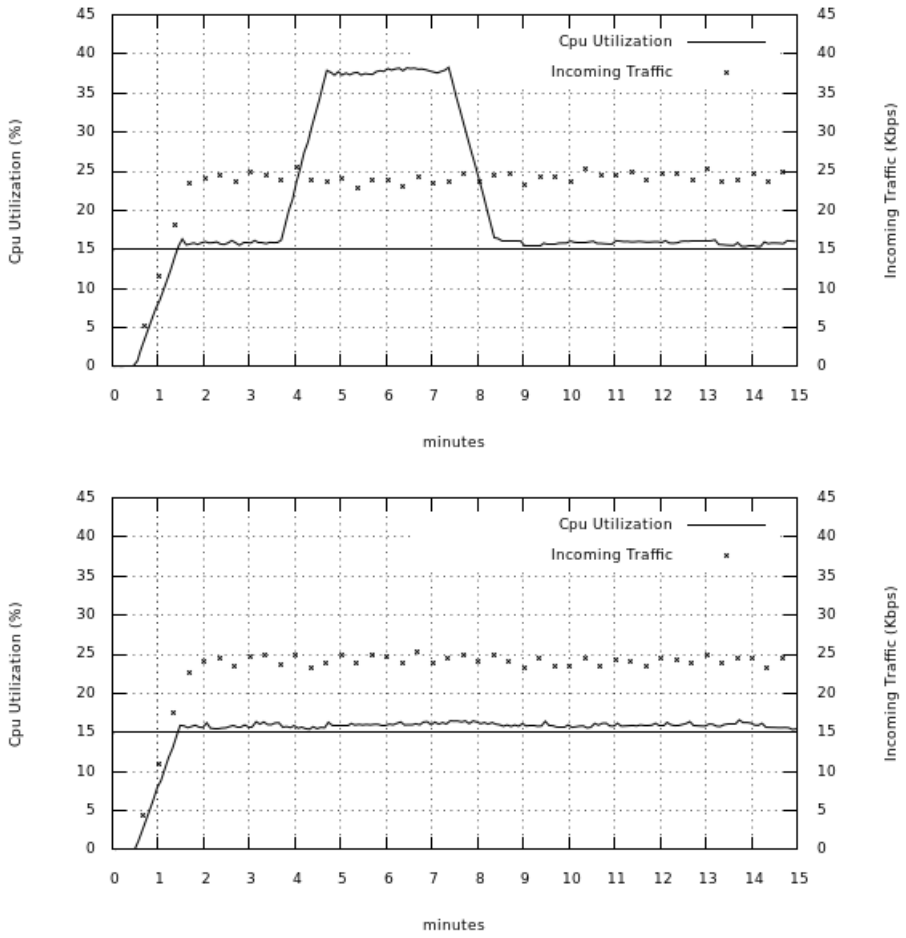


Figure 3. CPU utilization and network traffic on Server 0 (top) and Server (bottom) under modificed round robin policy for the infinite load scenario. At time $t = 4$ minutes high priority demand arrives in Server 0 that lasts until $t = 7$ minutes.

note that the delay between requests leaving the client increases (recall that client run an infinite request loop). At around time $t = 10$ the delay between requests return to original values. Thus, the proposed controller is effectively shielding the servers, pushing back to the client the dynamic adjustments in the request rates.

4.2. Limited load scenario

In this scenario the client application generates low priority requests periodically (at a fixed rate) such that the CPU load imposed on the servers is below the target of 15% (when servers are idle). However, this CPU load surpass the target when high priority demands arrives to the servers. Again, at around $t = 3$ minutes, high priority demand arrives to Server 0, generating a CPU load of about 25%, which finishes at around $t = 8$ minutes.

Figure 6 shows the behavior of the round robin policy, which as expected does not react to the sudden increase in load. The policy continues to dispatch requests to both servers equally, demanding CPU resources from Server 0, and thus reducing the CPU available to process the high priority demand.

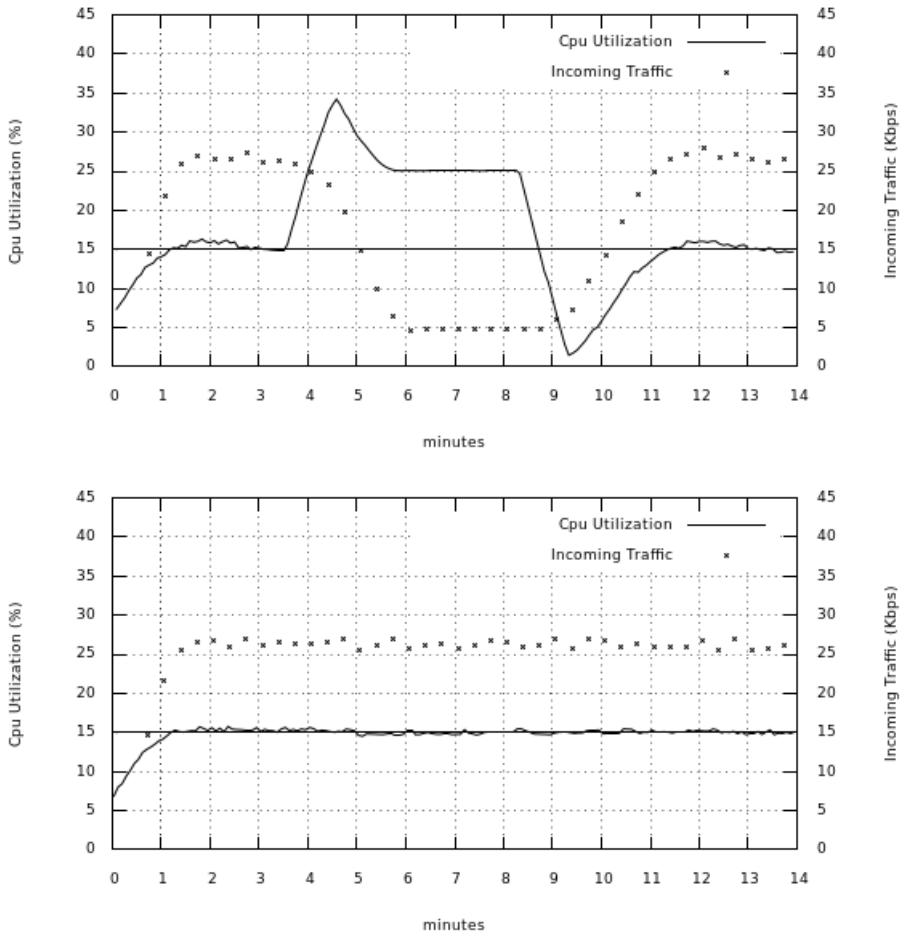


Figure 4. CPU utilization and network traffic on Server 0 (top) and Server (bottom) under proposed controller for the infinite load scenario. At time $t = 4$ minutes high priority demand arrives in Server 0 that lasts until $t = 7$ minutes.

As expected, the proposed mechanism behaves quite differently, as shown in Figure 6. In particular, the controller reacts to the increase in load in Server 0 by decreasing its low priority request rate, making available the needed CPU resources to process the high priority demand. Moreover, as the CPU utilization of Server 1 is still below the target, the load balancer increases the low priority request rate to that server. As a result, Server 1 absorbs the load that cannot be handled by Server 0, while meeting its target CPU utilization (of 15%). Note the increase in the network traffic to Server 1. At time $t = 9$ when high priority demand on Server 0 finishes, the controller again adapts and places part of the low priority request rate from Server 1 back to Server 0, equally distributing the load.

Interestingly, the reallocation of the client request rate from Server 0 to Server 1 allows the client to maintain its original request rate. Figure 8 shows that the client traffic remains unchanged while load is shiftted from Server 0 to Server 1 by the load balancer. The client is totally shielded from such dynamics on the servers, illustrating another benefit of the proposed controller.

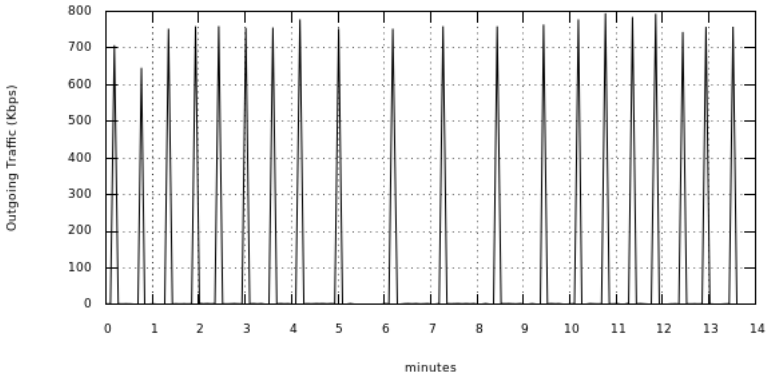


Figure 5. Outgoing network traffic in the client under the proposed controller for the infinite load scenario. Note that delay between requests increases at around $t = 4$ in response to high priority demands in Server 0.

5. Related Work

This paper considers the problem of distributing low priority requests across a set of servers while keeping CPU utilization imposed by such requests within a given target. This is necessary since servers must be ready to handle sudden high priority demands, not subject to the load balancer. Thus, our work is related to load balancing and to dynamic provisioning of resources, and in the following we comment on both research topics.

5.1. Load Balancing

Load balancing mechanisms are usually classified into static and dynamic. Static algorithms consider only nominal resources available on the servers, such as processing power and memory, but not any run-time characteristics, such as the actual load imposed on the server. The works of [Patel et al. 2013] and [Gandhi et al. 2015] present load balancing mechanisms to many different low-level contexts, such as NAT and Virtual IP Addresses. However, their proposal uses a simple static algorithm, known as *weighted random*.

The Weighted Least Connection (WLC) algorithm is a good example of dynamic load balancing since it sends arriving load to the server with smallest number of connections. However, this number may not accurately reflect the actual load on the server [Al Nuaimi et al. 2012]. Thus, [Ren et al. 2011] propose an improvement to the WLC algorithm, which considers a metric mixing usage of CPU, network, memory and disk. The algorithm then, having measured past values for such resources, predicts the next value for each server and sends the load to the one with the smallest value.

These algorithms are only marginally related to the our proposal, as they all consider a very different problem. In our scenario, we must decide not only which server is the best candidate for an arriving request, but also if any of the servers should receive the request at all! The servers may be too busy handling high priority demands and thus the load balancer may have to block the low priority requests. In other words, we perform not only load balancing, but also flow control. In this sense, our scenario and proposal also has similarities with dynamic resource provisioning.

[t]

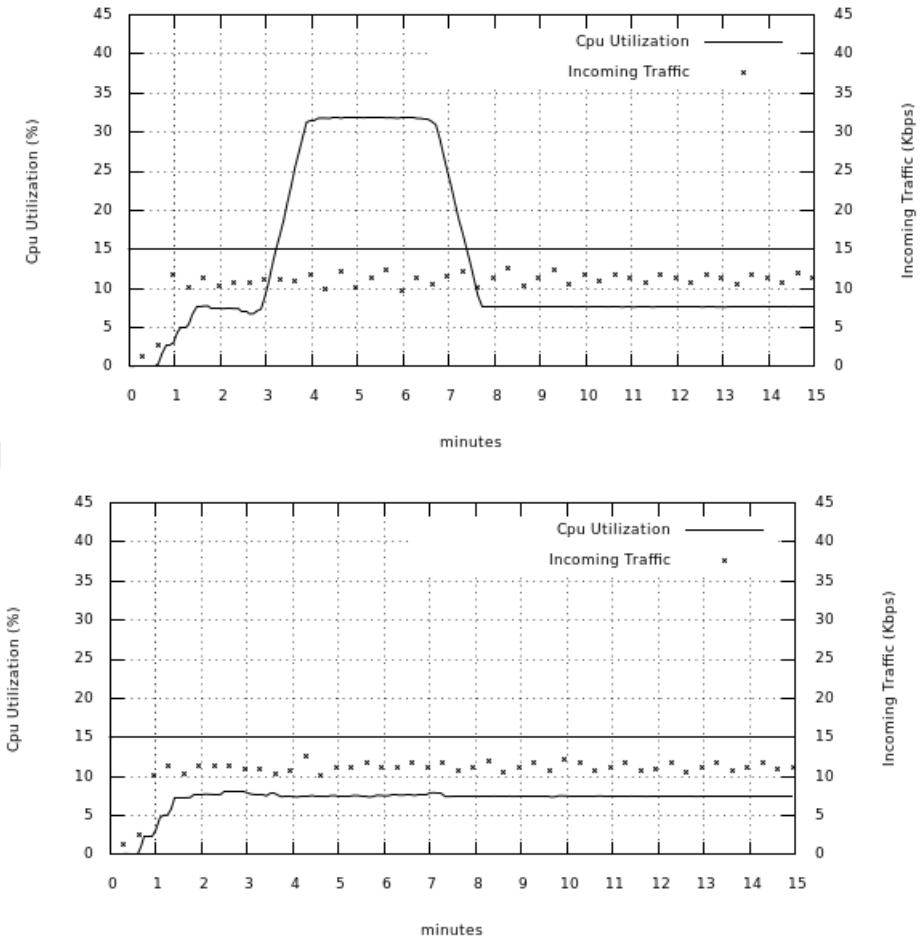


Figure 6. CPU utilization and network traffic on Server 0 (top) and Server (bottom) under modified round robin policy for the limited load scenario. At time $t = 4$ minutes high priority demand arrives in Server 0 that lasts until $t = 8$ minutes.

5.2. Dynamic Resource Provisioning

Dynamic resource provisioning is a highly researched subject nowadays, triggered mostly by large Cloud data centers and Big Data processing. In order for user applications to scale properly, more resources must be allocated to them when their load increases. However, the allocation should also be cost-effective and should not allocate more resources than needed. Thus, a lot of research has been done on automatic scaling techniques which can guarantee the SLA (Service Level Agreement) for the user applications cost-effectively for both the user and the Cloud provider. We comment on such mechanisms that leverage feedback/feedforward control methods, as our work also uses this methodology.

The work of [Dutreilh et al. 2010] mentions the difficulties of dynamic resource provisioning techniques based on thresholds or reinforcement learning when applied to real systems. They emphasize the fact that dynamic resource provisioning turn back to core concepts of automatic control - controllability, inertia, gain and stability. They summarize learned lessons mentioning good practices to be followed by automatic control algorithms - for example, that the time delay between two decisions must be long enough for the performance to stabilize to its new expected level, as we mentioned in Section 3.

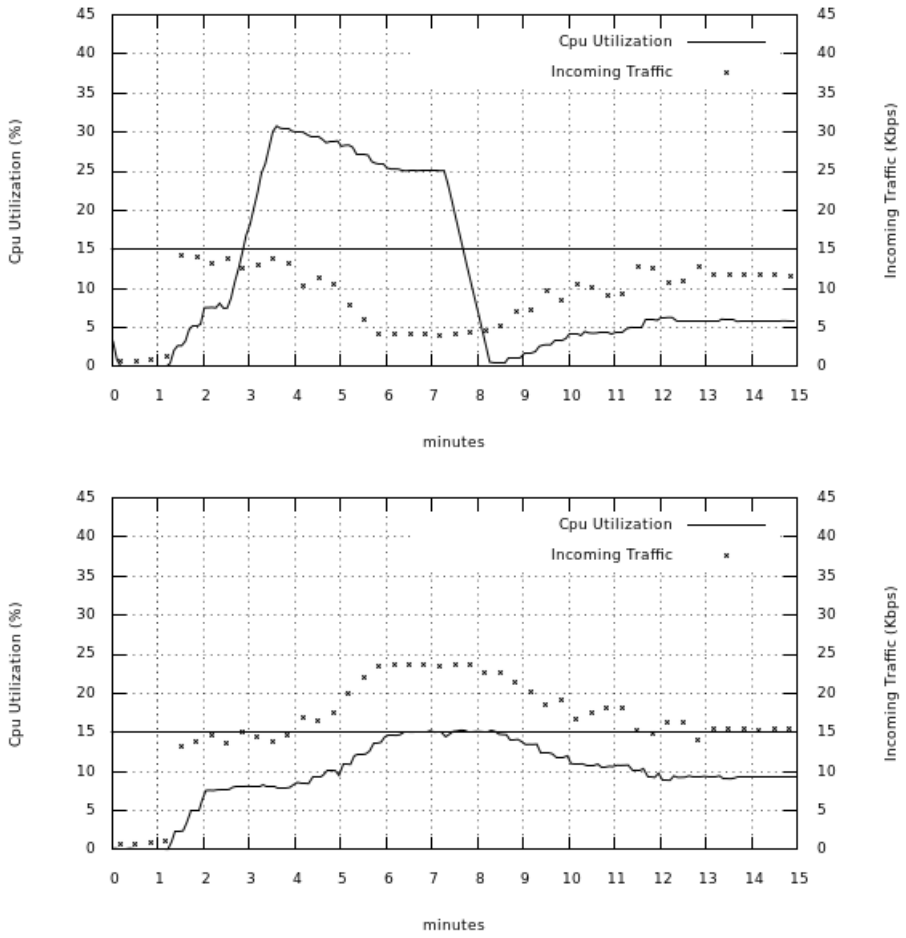


Figure 7. CPU utilization and network traffic on Server 0 (top) and Server (bottom) under the proposed controller for the limited load scenario. At time $t = 4$ minutes high priority demand arrives in Server 0 that lasts until $t = 8$ minutes.

Many papers have proposed dynamic provisioning algorithms for Cloud resources based on classical control theory. [Leontiou et al. 2010] propose a controller which uses a Proportional-Integral feedback and an adaptive feedforward methods to guarantee the SLA of general Cloud services. [Xiong et al. 2011] propose an adaptive PI controller to control the mean round trip time for N-tier web applications. [Ashraf et al. 2012] propose a proportional-derivative algorithm for scaling the application server tier and have deployed a prototype implementation in the Amazon Elastic Compute Cloud. The work of [Al-Shishtawy and Vlassov 2013] proposes a controller which mixes classical PI feedback with a feedforward controller to scale horizontally a cluster of key-value datastore and evaluate their algorithm on a Voldemort deployment.

Other researches, while not proposing classical control algorithms, have also based their proposal on feedback control theory. The work of [Zhu and Agrawal] proposes an algorithm based on an adaptive feedback controller for the allocation of cloud resources. The work of [Berekmeri et al. 2016] proposes a control algorithm based on classic feedback and feedforward methods to scale automatically the number of nodes in a MapReduce cluster in order to guarantee a certain level of performance.

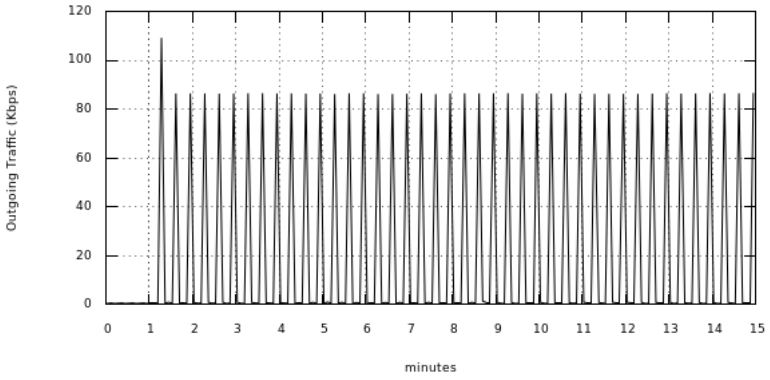


Figure 8. Outgoing network traffic in the client under the proposed controller for the limited load scenario. Note that client request rate remains unchanged despite the significant decrease in the request rate to Server 0.

The papers above assume there is always more available resources that can be allocated to an application in order to maintain its SLA. This is a reasonable assumption in many cases. However, our problem considers a small pool of resources which must be shared between applications with different priorities, and when a resource is allocated to handle a request, it immediatly interferes with the other requests. It is also worth noting in our scenario only low priority demands are subject to control, with high priority demands arising arbitrarily to servers.

6. Conclusions

This paper presented a load balancing mechanism based on closed-loop controller for a scenario where low priority requests traverse the load balancer while high priority demands can suddenly arrive directly at the servers. In particular, beyond distributing the load among the servers, the load balancer keeps CPU utilization arising from low priority requests within a specified target.

A real implementation of the controller has been developed and used to evaluate two different scenarios that illustrate that typical load balancing policies, such as round robin, are not appropriate in this context. In contrast, the proposed mechanism was shown to successfully adapt to dynamic changes to high priority demands, reducing or increasing the load imposed by low priority requests, meeting the target CPU utilization.

In the specific scenario considered, CPU utilization was measured by a one-minute moving average and thus the controller response had timing parameters of the same order. In some situations, a one-minute time constant may not be sufficient, but that is not a limitation of the proposed load balancing mechanism. If CPU utilization is measured using a smaller time-window, then it is possible to obtain a controller with faster response using the same design we proposed, by simply properly adjusting the time constants.

Finally, as the design of the controller used by the balancer needs tuning the value of some constants, a future work is to automatically determine these constants using an online regression with recursive least squares for example, establishing a relationship between the variation of measured feedback values to the variation of the control signal.

References

Al Nuaimi, K., Mohamed, N., Al Nuaimi, M., and Al-Jaroodi, J. (2012). A survey of load balancing in cloud computing: Challenges and algorithms. In *Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on*, pages 137–142. IEEE.

Al-Shishtawy, A. and Vlassov, V. (2013). Elastman: autonomic elasticity manager for cloud-based key-value stores. In *Proceedings of the 22nd international symposium on High-performance parallel and distributed computing*, pages 115–116. ACM.

Ashraf, A., Byholm, B., Lehtinen, J., and Porres, I. (2012). Feedback control algorithms to deploy and scale multiple web applications per virtual machine. In *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, pages 431–438. IEEE.

Berekmeri, M., Serrano, D., Bouchenak, S., Marchand, N., and Robu, B. (2016). Feedback autonomic provisioning for guaranteeing performance in mapreduce systems. *IEEE Transactions on Cloud Computing*.

Dutreilh, X., Moreau, A., Malenfant, J., Rivierre, N., and Truck, I. (2010). From data center resource allocation to control theory and back. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 410–417. IEEE.

Gandhi, R., Liu, H. H., Hu, Y. C., Lu, G., Padhye, J., Yuan, L., and Zhang, M. (2015). Duet: Cloud scale load balancing with hardware and software. *ACM SIGCOMM Computer Communication Review*, 44(4):27–38.

Leontiou, N., Dechouniotis, D., and Denazis, S. (2010). Adaptive admission control of distributed cloud services. In *2010 International Conference on Network and Service Management*, pages 318–321. IEEE.

Patel, P., Bansal, D., Yuan, L., Murthy, A., Greenberg, A., Maltz, D. A., Kern, R., Kumar, H., Zikos, M., Wu, H., et al. (2013). Ananta: cloud scale load balancing. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 207–218. ACM.

Ren, X., Lin, R., and Zou, H. (2011). A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast. In *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, pages 220–224. IEEE.

Xiong, P., Wang, Z., Malkowski, S., Wang, Q., Jayasinghe, D., and Pu, C. (2011). Economical and robust provisioning of n-tier cloud workloads: A multi-level control approach. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 571–580. IEEE.

Zhu, Q. and Agrawal, G. Resource provisioning with budget constraints for adaptive applications in cloud environments. *IEEE Transactions on Services Computing*, 5(4):497–511.

MALiBU: Metaheuristics Approach for Online Load Balancing in MapReduce with Skewed Data Input

Matheus H. M. Pericini[1], Lucas G. M. Leite[1] and Javam C. Machado[1]

[1]LSBD – Departamento de Computação – Universidade Federal do Ceará (UFC)
Campus do Pici – Bloco 952 – 60.020-181 – Fortaleza – CE – Brazil

{matheus.pericini,lucas.goncalves,javam.machado}@lsbd.ufc.br

Abstract. *MapReduce is a parallel computing model where a large dataset is split into smaller parts and executed on multiple machines. When data are not uniformly distributed, we have the so called partitioning skew, where the allocation of tasks to machines becomes unbalanced, either by the distribution function splitting the dataset unevenly or because a part of the data is more complex and requires greater computing effort. To solve this problem, we propose a function based on Simulated Annealing metaheuristic which finds a partitioning that results in a better load balancing.*

1. Introduction

As technologies related to computational clouds, sensors, and scientific computing advance, larger volumes of data are generated out of those applications, demanding efficient techniques to store and analyze such data in order to extract relevant information from them. To meet this need, the MapReduce model was created as a parallel computing model that distributes a large dataset to run on multiple commodity machines, avoiding the need of using machines with high processing power, making processing more robust and scalable. The standard structure of MapReduce is divided in two steps: Map and Reduce. In the Map phase, the master node takes an input, splits it into smaller subproblems, and distributes it to worker nodes. A worker node processes its subproblem creating intermediate data, stores that data and signals the master node. In the Reduce phase, the master node distributes the intermediate data to the workers according to a partition rule. The workers collect the intermediate data, combine and process them to form an output to the original problem. This divide-and-conquer process allows MapReduce to process large data sets on distributed clusters. A MapReduce user only needs to write the map and reduce functions, which effectively hides the operation of large clusters, providing a highly scalable and fault-tolerant solution for large data-processing applications. Many companies, such as Google, Amazon, Facebook, and Yahoo! have been using MapReduce to process large volumes of data [Zhihong Liu 2016].

Although MapReduce has several advantages, its simplicity may imply in difficulties when processing specific applications or specific data. In most studies involving MapReduce, it is considered that the data to be processed are balanced, so that its distribution between the worker nodes using a partitioning function based on its hash code is balanced in an acceptable way. However, in many real-world applications, data split between worker machines is not uniformly distributed and the partitioning function based on hash can not guarantee that the data allocation to the worker nodes will be balanced. This phenomenon is called Partitioning Skew, which is a problem that occurs when the

distribution of the data among machines is unbalanced or a part of the data needs more computational power to execute. As a result, in the reduce step, some of the workers become overloaded while others become idle.

The load balancing problem with skewed data has been investigated in several recent works [Ramakrishnan et al. 2012, Gufler et al. 2011, Kwon 2012]. Most approaches treat the problem offline, that is, wait for the mapping phase to complete to obtain the statistics related to the generated intermediate data, or perform a preprocessing on a sample of the data prior to the map phase to estimate the distribution of the data and execute an appropriate allocation strategy. However, these approaches lead to a network overhead and I/O excessive costs. Another strategy is to reallocate the data that requires more computational power to machines with more resources [Kwon 2012]. Although this strategy improves the performance of large data, it generates a large overhead, mainly when one considers small amount of data.

In this article, we balance the data by replacing the default MapReduce partitioning function with a function that also considers both the load already allocated to each machine and the size of the data that is waiting for allocation. Our strategy is to minimize a function that measures the imbalance of the load allocated to each machine. To find a partition of the data that approaches the ideal balancing, we will use a strategy based on metaheuristics because we want to solve this problem in an online manner without prior knowledge about the behavior of the data and it would also be computationally expensive to explore the entire solution space. Our strategy, called *MALiBU*, is based on Simulated Annealing metaheuristic, which is able to find solutions close to the global optimum while having relatively small complexity.

The remainder of this paper is organized as follows. Section 2 will give a more detailed view of the MapReduce model and the partitioning skew problem, motivating this work. Section 3 describes some of the related works in the literature. In Section 4 our contribution is presented, detailing MALiBU the partitioning algorithms used. Finally, Section 5 will show the results of the experimental evaluation and Section 6 will present our conclusions on this work.

2. MapReduce and Partitioning Skew

MapReduce splits dataset into smaller parts in order to process them on multiple machines [Dean and Ghemawat 2008]. The results of processing these parts are aggregated in order to generate the solution to the original dataset. In MapReduce the map tasks are responsible for receiving parts of the dataset, applying a user-defined mapping function and returning a set of key-value pairs, called intermediate data. The intermediate data are then distributed to the reduce tasks according to a partitioning function. A reduce task receives a partition and performs a reduce function to generate the solution for the intermediate data in this partition. MapReduce is able to significantly reduce the processing time of a large amount of data because it divides these data into smaller parts and processes them in parallel across multiple machines.

The MapReduce model was written in different programming languages [Isard et al. 2007, He et al. 2008, Ranger et al. 2007]. For example, Apache Hadoop [Hadoop 2006], one of its most popular free implementations, is based on a master-worker architecture, where a master node makes scheduling decisions and multiple worker nodes

run tasks dispatched from the master.

Figure 1 shows the architecture of the Hadoop MapReduce model.

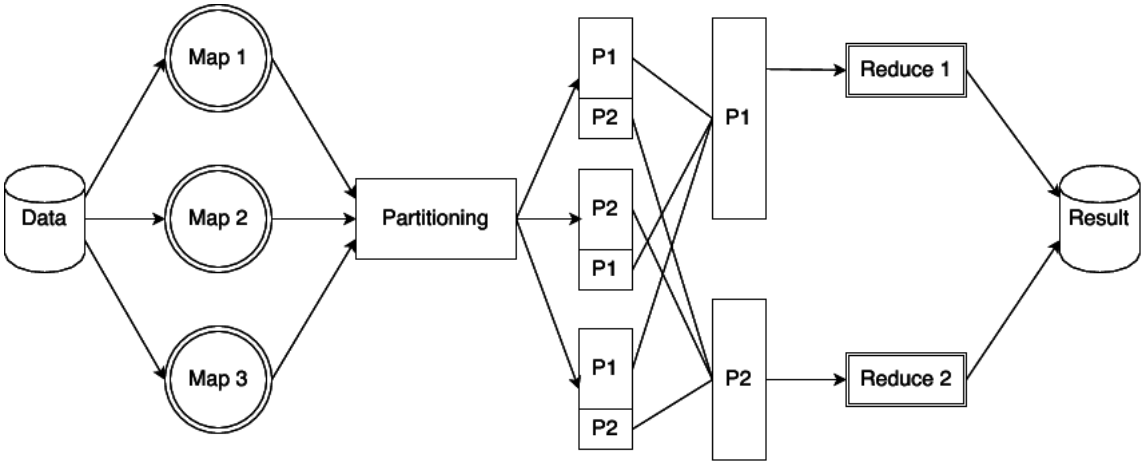


Figure 1. Hadoop MapReduce architecture

2.1. Partitioning Skew

Currently, in MapReduce implementations, data are allocated to the reduce nodes according to the function $Hash(HashCode\ mod\ number\ of\ reducers)$. Usually this function allocates tasks so that the partitions get balanced. When the size of the key-value pairs are not uniformly distributed, the allocation of tasks becomes unbalanced. This type of situation characterizes partitioning skew, in which some reduce nodes receive high loads, increasing the time of response of assigned tasks, and others receive low loads, being idle because they have more resources allocated than necessary. Many authors studied the partitioning skew in mapreduce applications [Kwon et al. 2011, Okcan and Riedewald 2011, Ibrahim et al. 2010, Atta et al. 2011].

2.1.1. Skewed Key Frequencies

For a number of applications, some keys occur more frequently in the intermediate data. For example, in word count some words appear more frequently than others, such as definite articles. In this type of application, the default Hadoop partitioning function will allocate tasks overloading the reduce nodes that receive most keys.

In the Figure 2, the partitioning skew generated by skewed key frequencies is illustrated. Notice that the K1 partition has got more intermediate data allocated than the others. If the key-value pairs (k1, v1) and (k2, v2) were respectively allocated to partitions K2 and K3, the partitions would be balanced.

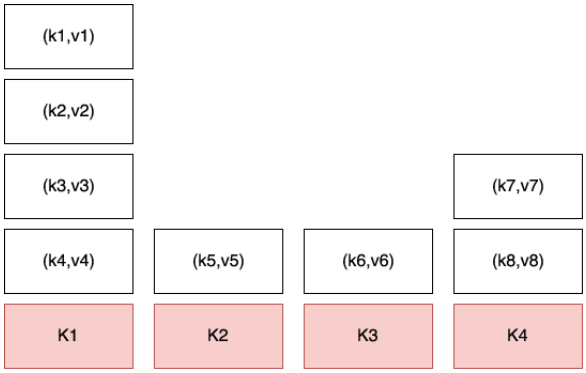


Figure 2. Skewed key frequencies example

2.1.2. Skewed Key Size

In applications where value sizes in key-value pairs vary significantly, the distribution of the tasks to reduce nodes may become unbalanced. Thus, although the number of different values for each key is approximately the same, the execution time may vary between one key and another, resulting in some idle nodes while others are overloaded.

In the Figure 3, the partitioning skew generated by skewed key sizes is shown. Although only one key value data is allocated to partition K3, this data takes longer to execute than the two data allocated to partition K2. In this case, for the nodes to be idle for the shortest possible interval, it is necessary that the data whose value is v4, allocated to partition K1 be allocated to partition K2 or K4.

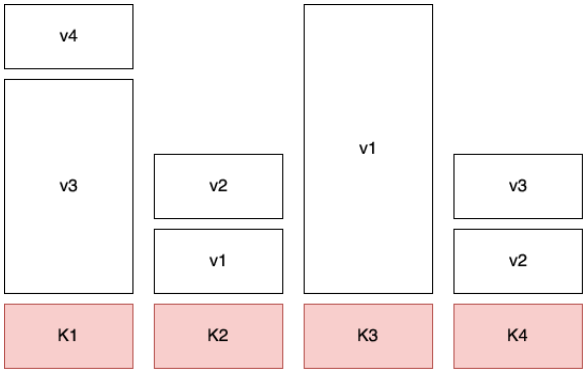


Figure 3. Skewed key size example

2.1.3. Skewed Execution Times

Some tasks may require more computing power than others. An example where this can occur is in the page classification algorithm, that classifies a page according to the links to other pages, which we call neighboring pages. The greater the number of neighbors pointing to the same page, the longer it will take to process it. In Figure 4, 128 machines processes a set of pages. During execution a machine takes about five times more than the average of all others.

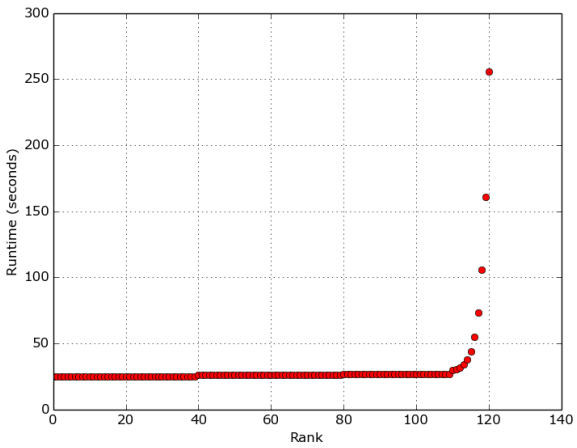


Figure 4. Skewed execution times example

3. Related Work

MALiBU focuses on the partitioning skew caused by skewed key frequency and skewed key size, since these types of skew are the major factors impacting job execution time. Recent works [Gufler et al. 2012, Ramakrishnan et al. 2012] and [Zaheilas and Kalogeraki 2014] show that solving the partitioning skew is not a trivial task. Many other works proposed solutions for the partitioning skew [Rosen and Zhao 2012], we detail the most important ones in this section.

3.1. OPTIMA: On-Line Partitioning Skew Mitigation for MapReduce with Resource Adjustment

OPTIMA [Zhihong Liu 2016] is a resource allocation framework based on the detection of overloaded reduce tasks due to partitioning skew. The approach sends information about the allocation of the intermediate data to the controller machine, which uses it to predict the size of tasks and identify possible overloaded tasks. With this information, OPTIMA allocates resources for overloaded machines up to the maximum machine load limit. OPTIMA implements five main components. During the execution of the map tasks, the Partition Size Monitor sends the size of the intermediate data produced to the master machine through a modification in the reporter responsible for updating the master. With the information gathered from the map tasks, Partition Size Predictor uses a linear regression to predict the size of the partitions in an online manner, so that Overloaded Task Detector be able to identify the reduce jobs that are overloaded and tells the resource allocator that these tasks will need more performance. Lastly, the Fine-grained Container Scheduler increases the amount of memory and CPU used by machines that need higher performance to the maximum amount available on the machine based on information about overloaded tasks. Despite the decrease of a job execution time, OPTIMA is not able to prevent a machine from draining its resources and become overloaded because it continues to use the standard partitioning function.

3.2. SkewTune: Mitigating Skew in MapReduce Applications

Skewtune [Kwon 2012] monitors jobs in order to identify which ones are responsible for overloading reduce nodes. With this information, it balances the utilization of the

cluster nodes by repartitioning these jobs, allocating them to other reducers with less load. The process consists in verifying if a job that is causing the overhead is at a point of its execution in which it is advantageous to divide its data with a reducer that is underloaded. This division will only take place if the time needed to divide the data plus the estimated time remaining for the split job is less than the remaining time of the complete job. This process is repeated until all tasks are completed or there is no task that can be divided. At the end of the execution, the data of the same task that has been divided are grouped in the same machine, in order to keep the result consistency of MapReduce. Although it works well for complex tasks, overhead caused by the analysis and division of a small task generates an additional cost that ends up negatively impacting the execution time.

3.3. Online Load Balancing for MapReduce with Skewed Data Input

The work described in [Yanfang Le 2014] presents two key allocating algorithms for the partitioning phase. The first is a greedy online algorithm that acts on the partitioner distributing the keys according to the size of the existing partitions giving priority to the smaller ones. In the second algorithm, it performs the distribution of the keys based on a preprocessing over the statistics of a priori data. Both algorithms ignore that the key distribution can vary significantly during a mapping phase, causing an imbalance in the allocation of the processed keys, failing to avoid partitioning skew in a significant way.

The approaches described as related work show different ways of dealing with partitioning skew. In the next section we present MALiBU, a meta-heuristic approach to balance the load allocated to the reduce nodes, avoiding the overload of these nodes without generating a large overhead in the reduce phase.

4. The MALiBU Method

MALiBU balances the load received by the reduce nodes following a combinatorial optimization problem solution. We assume that the input of this problem is the generated intermediate data set and the output is a partitioning of the keys of this set into a number of reducers. Because it is an online solution involving a large amount of data, we apply metaheuristic approaches to find a sufficiently good solution. In combinatorial optimization, by searching over a set of feasible solutions, metaheuristics can often find good solutions with less computational effort than optimization algorithms, iterative methods, or simple heuristics [Blum and Roli 2003]. Metaheuristics use a sample of the solution set, which is too large to be completely covered. MALiBU takes advantage of the Simulated Annealing metaheuristic to partition the keys to reduce tasks. In a metaheuristic, we look for a solution from small variations of a candidate solution, which can be generated in a random way. This variation is compared with the candidate solution for evaluation according to some criterion of acceptance. The best evaluated solution is maintained and the process is repeated until a stopping criterion is satisfied. In our approach, a candidate solution is a variation of a partitioning which we will call its neighboring partition. A partition A is a neighbor of a partition B if they differ by the allocation of exactly one key. To facilitate the understanding of the metaheuristics, we define below the formal concept of neighboring partitions and detail later the operation of Simulated Annealing in the partitioning phase of MapReduce.

Definition 1 (Neighboring partitions)**.** Let $K = \{k_1, k_2, \ldots, k_n\}$ be a set of keys, $R = \{r_1, r_2, \ldots, r_m\}$ a set of reducers and $P_1 = \{p_1, p_2, \ldots, p_m\}$, $P_2 = \{p_1, p_2, \ldots, p_m\}$ two partitions from K to R. We say that P_1 is a neighbour of P_2 if and only if just one of the following conditions is satisfied:

- $k_i \in p_x \in P_1$, $k_j \in p_y \in P_1$, $k_j \in p_x \in P_2$ and $k_i \in p_y \in P_2$, with $1 \leq i < j \leq n$ and $1 \leq x < y \leq m$ for only one pair of keys k_i and k_j. The partitioning is the same for the remaining keys in P_1 and P_2
- $k_i \in p_x \in P_1$ and $k_i \in p_y \in P_2$, with $1 \leq i \leq n$ and $1 \leq x < y \leq m$ for only one key k_i. The partitioning is the same for the remaining keys in P_1 and P_2

4.1. Simulated Annealing

In metallurgy, annealing is a process for hardening metals or glass by heating and cooling them gradually, allowing the material to reach a low-energy crystalline state [Russell et al. 2003]. The Simulated Annealing (SA) can be seen as the process of making a ping-pong ball reach the bottom of an irregular surface. If we just let the ball roll, it can get stuck at a point other than the bottom, because the surface is uneven. So that the ball can leave the current point and continue rolling, it is necessary for someone to shake the surface.

In our problem, we want to attenuate the imbalance of the intermediate data allocated to the reducers. The solution based on SA is done as follows. An initial temperature and a cooling factor are defined. According to the intermediate data given as input, a key partitioning is created using the standard Hadoop partitioning function. We call this partitioning *current partitioning*. From this, we compare the current with a randomly generated neighbor partitioning. If the neighboring partitioning better balances the intermediate data allocated to each reducer, it becomes the current partitioning. Otherwise, the neighboring partitioning will become the current partitioning with a probability less than 1. This probability decreases proportionally to the current temperature. Finally, the current temperature is updated according to the cooling factor. This process is repeated until the temperature reaches 0. To facilitate understanding, we will show the pseudocode of this procedure.

In the Algorithm 1, we aim to reduce the imbalance, which is the load of the reducer with the highest volume of data allocated. That is, in order to balance the load, we must partition the keys between the reducers so that the allocated load of the machine with the highest allocation is minimal.

Our implementation consists of integrating Simulated Annealing to Hadoop, using it as the partitioning function. The partitioning phase works as follows: as the dataset is processed and the intermediate data is generated, Hadoop accumulates a portion of that data and Simulated Annealing is executed to find an allocation that minimizes the unbalance of the reducers. This process will repeat itself until the end of the map phase.

5. Experimental Results

In this section we compare the performance of MALiBU against the standard partitioning function of Hadoop YARN 2.7.2. For the implementation of the described algorithms, we modified the Hadoop YARN 2.7.2 by adding the partitioning functions and bypassing the native partitioning function.

Algorithm 1 Simulated Annealing

procedure PARTITIONER(D,T,C,R)
  $current \leftarrow$ partitioning of the keys of D using $Hash(HashCode(D)\ mod\ |R|)$.
  $t \leftarrow T$
  $c \leftarrow C$
  while $T \neq 0$ **do**
    $neighbor \leftarrow generateNeighbor(current)$
    $b_1 \leftarrow unbalance(current)$
    $b_2 \leftarrow unbalance(neighbor)$
    if $b_2 < b_1$ **then**
      $current \leftarrow neighbor$
    else if $random(0,1) < e^{\frac{b_1 - b_2}{t}}$ **then**
      $current \leftarrow neighbor$
    $t = t - c$
return $current$

Our experiments were done using Hadoop YARN 2.7.2 with a master node and nine worker nodes with 8GB of RAM, 4 virtualized CPUs and 80GB of hard drive. All virtual machines are running in the OpenStack cloud environment hosted at the LSBD/UFC. We executed a WordCount application over a real dataset made up of book reviews from the Amazon store [McAuley et al. 2015b, McAuley et al. 2015a]. This dataset has an approximate size of 500MB.

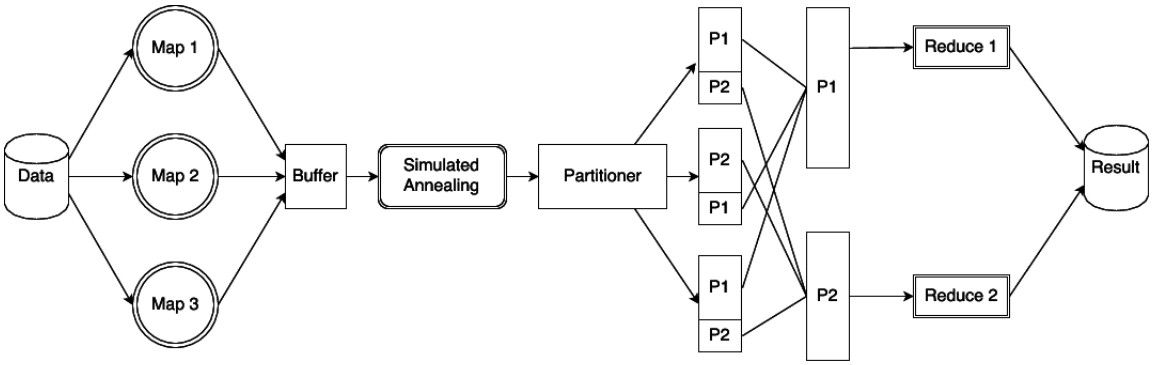


Figure 5. MALiBU architecture

The MALiBU execution occurs as follows: the first $X\%$ of the intermediate data generated in the map phase are accumulated in a buffer. Once the processing is done at $X\%$ percent, this data is partitioned according to Simulated Annealing (Figure 5), which is set to an initial temperature of Y and a cooling factor of 1. After that, each arriving key is either allocated according to the Simulated Annealing result or it is regularly allocated if it's a new key. This process repeats until all the intermediate data is generated and allocated to its due partitions, which in turn, will be executed by the reduce nodes of the system. For our experiments, we consider $X \in \{5, 10, 15, 20\}$ and $Y \in \{5000, 10000, 20000, 30000\}$. The complete MALiBU architecture can be seen in Figure 5.

Next, we check the performance of the native Hadoop partitioning functions and

MALiBU for different scenarios. To measure the performance, we calculate the size of the partition that has the largest amount of data allocated (Y axis) taking into account the progress of the data map phase (X axis).

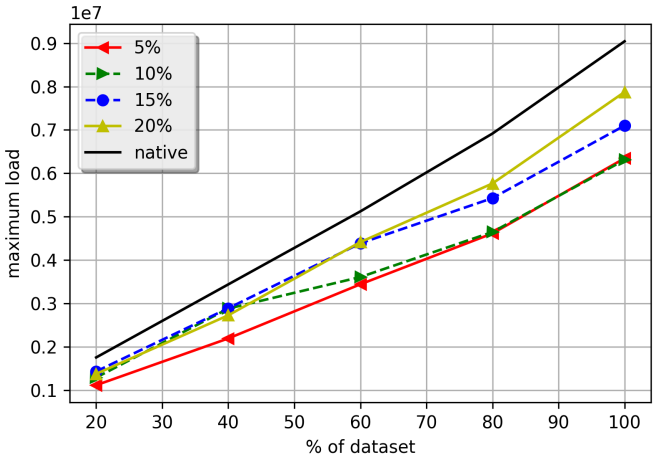


Figure 6. Comparison of several scenarios of data accumulation of the proposed method using 5000 of temperature with the native Hadoop partitioning function.

In our first scenario, as seen in Figure 6, we set a temperature of 5,000 and we varied the amount of data accumulated. Our results showed an improvement in the data balancing in relation to the native Hadoop solution, which is represented by the black line in Figure 6. As expected, the solutions generated by approaches that accumulate less data achieved better results, since the temperature of 5,000 is small in relation to the amount of accumulated data.

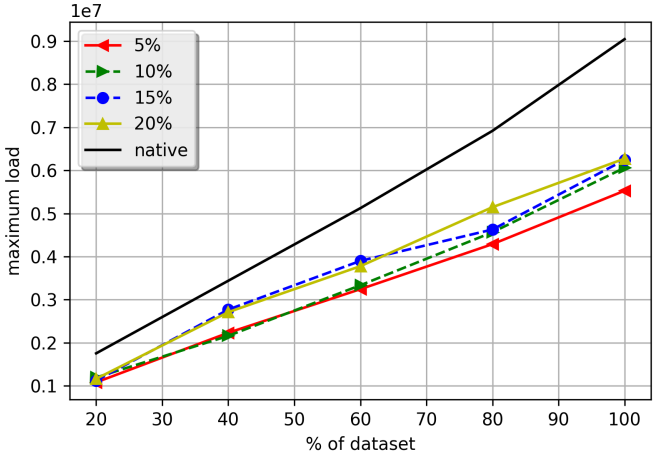


Figure 7. Comparison of several scenarios of data accumulation of the proposed method using 10,000 of temperature with the native Hadoop partitioning function.

In the next scenario, as seen in Figure 7, we set a temperature of 10,000 and we varied the amount of data accumulated. Considering the previous scenario, we have

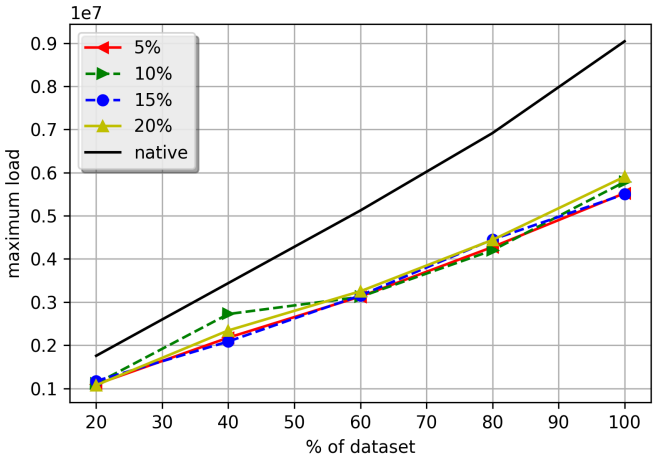


Figure 8. Comparison of several scenarios of data accumulation of the proposed method using 20000 of temperature with the native Hadoop partitioning function.

achieved an improvement in all data accumulation approaches, showing that the increase in temperature reflects in fact a better allocation of the data.

In the next scenario, described in Figure 8, we set a temperature of 20,000 and we varied the amount of data accumulated. Our results showed a slight drop in the maximum load allocated to a reduce node in relation to the temperature set at 10,000. In addition, we notice that as data accumulation approaches show a small difference between them, we can say that the temperature is generating a result close to optimal in relation to the accumulated data quantities.

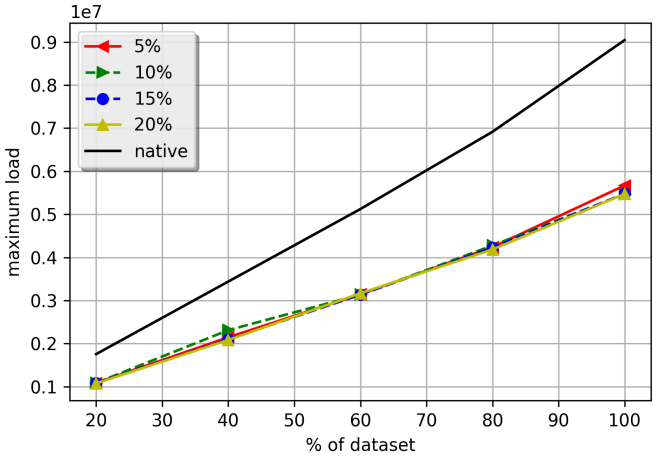


Figure 9. Comparison of several scenarios of data accumulation of the proposed method using 30000 of temperature with the native Hadoop partitioning function.

In our last scenario (Figure 9), we set the temperature of 30,000 and we varied the amount of data accumulated. The results showed a slight improvement in relation to the results presented in the previous scenario, where we set the temperature at 20,000. This result indicates a convergence, so that more temperature or more accumulated data

will not be able to generate a better distribution, due to the fact that the data allocation is already very close to the optimal allocation for the dataset used.

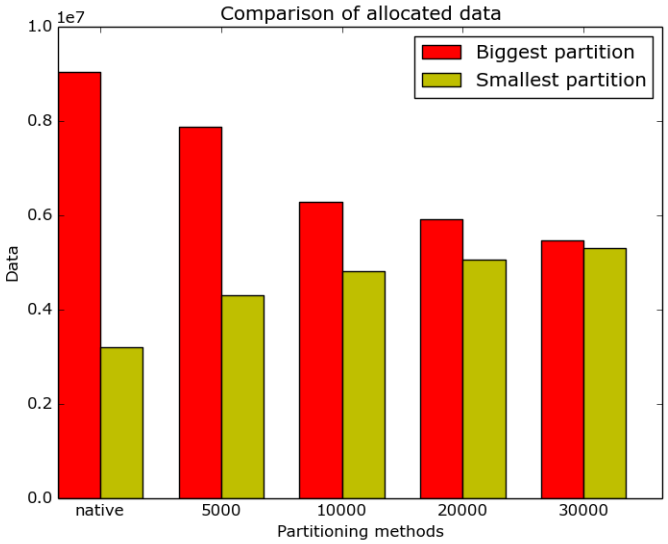


Figure 10. Comparison of the load allocated to smaller and larger partition in different scenarios.

In our experiments, as seen in Figure 10, each time MALiBU accumulates 20% of data, it returns partitions more balanced than all the other scenarios studied. It is worth mentioning that the comparative approaches have been used against data that results in partitioning skew, which is the type of dataset that this work proposes to deal with.

6. Conclusion

In this paper, we investigated the performance of the MapReduce model in a scenario where partitioning skew occurs. This scenario motivated us to mature a balancing approach of the data allocated to the reducers. By verifying the points in which the standard MapReduce could be modified so that partitioning skew was attenuated, it was noticed that altering the partitioning function would be beneficial for the allocation of the data to the reducers, improving data balancing among nodes.

We developed MALiBU, a partitioning method that uses the Simulated Annealing metaheuristics to minimize the imbalance of data on the machines involved in MapReduce. The use of a metaheuristic is interesting because it prevents the space of solutions to be completely covered, focusing on the most promising solutions. We proposed our approach using Simulated Annealing because it is a metaheuristic that is able to converge to a global optimal as long as the chosen temperature matches the complexity of the function to be optimized. Hill-Climbing-based metaheuristics may cling to local optima and may return unsatisfactory results. In addition, Simulated Annealing has a relatively low computational cost, favoring its use in an online algorithm. In our experiments using a 10-node cluster running an actual workload, we were able to improve data balancing performance reducing by 40% the size of the partition that has the largest amount of data allocated relative to the native Hadoop. When compared to other partitioning methods

in Hadoop, MALiBU is able to efficiently allocate the data to the reduce nodes. The counterpoint is that it becomes necessary to have sufficient computational resource available to use the temperature and data accumulation parameters for the dataset which the MapReduce application is intended.

Other authors have developed extensions to Hadoop. Among them, we can highlight [Yanfang Le 2014], which presented greedy partitioning method, which allocates the data to the partition with the least load. Although this solution finds good results with balanced data, it is not able to predict the evolution of the keys and, in this way, it can allocate two heavy keys in the same partition, compromising the load balancing in the reduction nodes. MALiBU circumvents this problem with a data accumulation strategy, similar to the sample-based approach described in [Yanfang Le 2014], running Simulated Annealing in order to analyze the evolution of each key size in a more efficient manner.

As future works, we intend to improve the key size prediction using machine learning methods, in order to reduce the amount of data accumulated and the temperature. With this, we can decrease the execution time of the method, making it closer to the execution time of the native Hadoop partitioning function. Another possibility of contribution is to use a heuristic that is able to find the ideal temperature for the execution of Simulated Annealing according to the amount of accumulated data, once the temperatures used in this work were chosen by experimentation. In this way, the system could avoid the absence or the wastage of resources when partitioning the accumulated data. In addition, we want to test the behavior of other metaheuristics, such as Local Beam Search and Stochastic Beam Search, in order to verify the difference in data allocation, memory consumption and runtime. Finally, we intend to study the partitioning skew generated by different execution times of the data. This type of skew comes from a different source and also requires a different kind of solution.

Acknowledgements

This research was partially supported by Lenovo/LIC, Capes/Brazil and LSBD/UFC.

References

Atta, F., Viglas, S. D., and Niazi, S. (2011). Sand join—a skew handling join algorithm for google's mapreduce framework. In *Multitopic Conference (INMIC), 2011 IEEE 14th International*, pages 170–175. IEEE.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308.

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Gufler, B., Augsten, N., Reiser, A., and Kemper, A. (2011). Handing data skew in mapreduce. In *Proceedings of the 1st International Conference on Cloud Computing and Services Science*, volume 146, pages 574–583.

Gufler, B., Augsten, N., Reiser, A., and Kemper, A. (2012). Load balancing in mapreduce based on scalable cardinality estimates. In *2012 IEEE 28th International Conference on Data Engineering*, pages 522–533. IEEE.

Hadoop (2006). [Online; accessed in 12-6-2016].

He, B., Fang, W., Luo, Q., Govindaraju, N. K., and Wang, T. (2008). Mars: a mapreduce framework on graphics processors. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 260–269. ACM.

Ibrahim, S., Jin, H., Lu, L., Wu, S., He, B., and Qi, L. (2010). Leen: Locality/fairness-aware key partitioning for mapreduce in the cloud. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 17–24. IEEE.

Isard, M., Budiu, M., Yu, Y., Birrell, A., and Fetterly, D. (2007). Dryad: distributed data-parallel programs from sequential building blocks. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 59–72. ACM.

Kwon, Y., B. M. H. B. R. J. (2012). Skewtune: mitigating skew in mapreduce applications. *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 25–36.

Kwon, Y., Balazinska, M., Howe, B., and Rolia, J. (2011). A study of skew in mapreduce applications. *Open Cirrus Summit*.

McAuley, J., Pandey, R., and Leskovec, J. (2015a). Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM.

McAuley, J., Targett, C., Shi, Q., and van den Hengel, A. (2015b). Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM.

Okcan, A. and Riedewald, M. (2011). Processing theta-joins using mapreduce. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 949–960. ACM.

Ramakrishnan, S. R., Swart, G., and Urmanov, A. (2012). Balancing reducer skew in mapreduce workloads using progressive sampling. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 16. ACM.

Ranger, C., Raghuraman, R., Penmetsa, A., Bradski, G., and Kozyrakis, C. (2007). Evaluating mapreduce for multi-core and multiprocessor systems. In *2007 IEEE 13th International Symposium on High Performance Computer Architecture*, pages 13–24. Ieee.

Rosen, J. and Zhao, B. (2012). Fine-grained micro-tasks for mapreduce skew-handling. *White Paper, University of Berkeley*.

Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., and Edwards, D. D. (2003). *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River.

Yanfang Le, Jiangchuan Liu, F. E. D. W. (2014). Online load balancing for mapreduce with skewed data input. *IEEE Conference on Computer Communications*.

Zaheilas, N. and Kalogeraki, V. (2014). Real-time scheduling of skewed mapreduce jobs in heterogeneous environments. In *11th International Conference on Autonomic Computing (ICAC 14)*, pages 189–200.

Zhihong Liu, Qi Zhang, R. B. Y. L. B. W. (2016). Optima: On-line partitioning skew mitigation for mapreduce with resource adjustment. *Journal of Network and Systems Management*.

Provendo Múltiplas Transferências de Dados em Massa com Roteamento e Alocação de Espectro Ciente da Aplicação em Redes Ópticas Elásticas

Léia Sousa de Sousa, André Costa Drummond

[1]Departamento de Ciência da Computação (CIC) – Universidade de Brasília (UnB)
Caixa Postal 4.466 – 70910-900 – Brasília – DF – Brasil

Abstract. *This paper presents three Application-Aware Routing and Spectrum Assignment (AARSA) for the Multiple Bulk Data Transfer (MBDT). Simulations were done in an inter-data center (IDC) network to compare the application-aware routing versus conventional routing, involving both resynchronization and backup applications. The results indicate that AA-RSA outperforms conventional RSA by providing up to* 70% *more of resynchronisations among data centers without increasing the blocking rate of backup requests.*

Resumo. *Este trabalho apresenta três soluções de roteamento e atribuição de espectro ciente da aplicação (AARSA) para Múltiplas Transferências de Dados em Massa (MBDT). Simulações foram realizadas em uma rede inter-centro de dados (ICD) para comparar o roteamento ciente da aplicação versus convencional, envolvendo-se aplicações de ressincronização e backup simultaneamente. Os resultados indicam que soluções cientes superam a convencional com ganho de até 70% no número de ressincronizações ICD efetivadas, sem impactar na taxa de bloqueio de backups.*

1. Introdução

Vários avanços em tecnologias de redes ópticas elásticas (EON) têm levado os provedores de nuvem a se prepararem para futura adoção em redes inter centros de dados (ICD) geo-distribuídas. A *Microsoft*, por exemplo, que atualmente utiliza tecnologia WDM, tem mostrado interesse em explorar o uso de *transponders* de largura de banda flexível (BVTs) e implementação de roteamento de alocação de espectro com modulação fixa (RSA), uma vez que sua infraestrutura é composta de uma variada gama de tipos de fibra ópticas, representando um desafio para o uso mais eficiente do espectro óptico [Filer et al. 2016].

A EON utiliza uma grade do espectro óptico com menores espaçamentos e transporta mais *bits* por símbolos, resultando em melhor ajuste das demandas às larguras de banda dos canais. Os BVTs da EON convertem o sinal vindo da rede cliente para um sinal a ser comutado no domínio óptico, e implementam técnicas *Orthogonal Frequency-Division Multiplexing*(OFDM) capazes de alocar fatias do espectro (*slots*) para um determinado caminho de acordo com a taxa de transmissão solicitada. Outros importantes elementos na EON são os *switches cross-connects* de largura de banda flexível (BV-WXCs), que estabelecem caminhos ópticos com os recursos solicitados por uma dada aplicação e podem suportar uma infinidade de caminhos com as mais variadas capacidades [Wan et al. 2012].

A principal preocupação dos provedores de serviços em nuvem é como lidar com o enorme volume de tráfego, que tem crescido 29% a cada ano, e como lidar com o crescente

período de tempo no qual a largura de banda permanece ocupada [Cisco 2016]. Nesse cenário, as aplicações de Transferências de Dados em Massa (BDT) e as múltiplas BDTs (MBDT), que pertencem à classe de tráfego de menor prioridade (classe *background*), acabam sendo penalizadas em detrimento de outras aplicações de maior prioridade, como as de vídeo IP. Isso ocorre porque BDTs e MBDTs movimentam volumosas quantidades de dados em redes ICD, são tolerantes a atrasos e podem esperar enquanto oportunidades são dadas ao tráfego que deve ser entregue com o menor retardo possível sob pena de arruinar a experiência dos usuários [Zhang et al. 2015].

As aplicações de transferências de dados mais comuns são: (i) os *backups*, transferências unitárias (BDT) correspondentes a um conjunto de vários tipos de dados que, por medidas de proteção, são diariamente encaminhados para CDs remotos e pertencentes a diferentes zonas de desastres (DZ), e (ii) as ressincronizações, que são MBDTs implementadas para atualizar grandes conjuntos de dados distribuídos, como forma de replicação para garantir tolerância a falhas. Essas aplicações podem ser favorecidas com a maior disponibilidade em EON, abrindo precedentes para a implantação de roteamento ciente da aplicação (*Application-Aware Routing*, AA-R), seguindo a tendência de adoção de elementos da rede de transporte capazes de distinguir fluxos e pacotes variados [Sadasivarao et al. 2016], graças a futura integração das tecnologias de rede de transporte óptica (OTN) e *rede definida por software* (SDN). O escalonamento de requisições é outra técnica que favorece o aproveitamento dos recursos da rede, permitindo que as requisições busquem por novas oportunidades de atendimento [Laoutaris et al. 2011].

Enquanto a sincronização requer que o nó detentor dos dados seja paralisado para que ocorra a transferência para todas as suas réplicas [Markowski 2016], na ressincronização segura, quando um lote de comunicação de muitos para um ($M \rightarrow 1, M \geq 3$) é estabelecido, mais nós são envolvidos no papel de emissor. Todas as operações são realizadas com os membros do grupo de replicação em franco funcionamento, e a quantidade de réplicas necessárias varia de acordo com o protocolo de replicação e o modelo de sistema adotado. Além disso, no modelo mais severo de falhas, as falhas Bizantinas, são necessárias 4 réplicas funcionando corretamente para que uma delas possa falhar [Castro and Liskov 2002]. Embora este número possa ser aumentado para permitir que mais réplicas falhem, na prática utilizam-se poucas réplicas [Vukolić 2010].

Este trabalho propõe soluções para o problema RSA, que são cientes da aplicação, e atendem aplicações de *backup* e ressincronização ICD. O principal objetivo é garantir que as ressincronizações, aplicações mais complexas, sejam efetuadas e façam uso mais eficiente dos recursos da rede, atendendo um número maior de conexões e aumentando o desempenho do sistema. Além disso, o escalonamento de chamadas será adotado para aumentar as chances de atendimento em momentos e oportunidades posteriores.

Os resultados obtidos indicam que o algoritmo ciente da aplicação tem taxa de sucesso de ressincronização até 70% maior do que com o uso de uma solução convencional. Além disso, o escalonamento de requisições permite que até 25% de todas as requisições que seriam bloqueadas por falta de banda, sejam atendidas em outros momentos, reduzindo as despesas com armazenamento em trânsito. Destaca-se ainda que, mesmo provendo ressincronizações com alto índice de sucesso, as operações de *backups* não foram penalizadas, uma vez que o percentual de bloqueio permaneceu entre 10% e 12%. Assim, as principais contribuições deste trabalho são: *(i)* proposta de três algoritmos RSA em

EON cientes da aplicação: AARSAE (sem escalonamento), MR (com escalonamento das ressincronizações) e MR+B (com escalonamento das ressincronizações e *backups*); e *(ii)* realização de simulações para avaliar o desempenho dos algoritmos propostos em comparação com uma solução RSA convencional, demonstrando claramente os ganhos advindos da exploração de informações das aplicações na rede.

2. Trabalhos Relacionados

Muitos dos grandes provedores de serviços em nuvem pretendem migrar suas tecnologias de rede de transporte atuais para a EON [Filer et al. 2016], motivados pela promessa de aumento nas taxas de transmissão de dados e utilização mais eficiente do espectro óptico [Wan et al. 2012]. Mas os benefícios irão além disso. As aplicações que são executadas nessa infraestrutura e os diferentes tipos de dados que nela transitam poderão ser identificados com especificidades pelos dispositivos da rede [Sadasivarao et al. 2016], e as que mais demandam recursos e representam custos significativos nas despesas dos CDs poderão ser beneficiadas com um roteamento mais inteligente, o AA-R. Outro tipo de roteamento inteligente é o roteamento ciente das limitações na camada física, que já é um tema tradicionalmente tratado em redes ópticas [Subramaniam et al. 2013].

Atualmente, as melhores soluções para lidar com o tráfego BDT e MBDT relativo a aplicações como *backups* e ressincronizações são baseadas em roteamento estático, envolvendo operações de escalonamento em diferentes janelas de tempo [Nandagopal and Puttaswamy 2012], mesmo quando canais comerciais de ISPs são utilizados de maneira associada à redes dedicadas [Li et al. 2012] e ainda, utilizando a largura de banda ociosa em qualquer ponto da rede, bem como apoiando-se em previsões de consumo de banda seguindo o histórico da rede [Laoutaris et al. 2011].

A partir do surgimento da EON, o atendimento de requisições orientadas a dados é tido como uma das soluções para aproveitar a fragmentação do espectro óptico [Lu et al. 2015b, Lu and Zhu 2015], no entanto, as aplicações dessa natureza ainda precisariam esperar por recurso suficiente, o que representaria fragmentações de tamanho considerável, ou continuariam alugando serviços de armazenamento ao longo da rede, aumentando o grau de vulnerabilidade dos dados. Por outro lado, [Wan et al. 2012] propõe duas soluções RSA convencionais, uma delas com modulação fixa e outra com modulação adaptativa a partir do nível mais eficiente para satisfazer a mais demandas.

Soluções de AA focadas em EON têm começado a ser exploradas na literatura, a exemplo de [Song et al. 2012], que propõe o mapeamento de máquinas virtuais aos seus respectivos *hosts* físicos para realizar migrações de acordo com a disponibilidade de recursos e com as distâncias entre esses *hosts*. Como resultado, reduziu-se o tráfego de dados nos enlaces da rede e o consumo de energia dentro do CD. Esse tema também tem começado a ser destacado no campo das SDNs [Zinner et al. 2014]. Em [Sousa et al. 2016] são mostradas as vantagens do roteamento ciente da aplicação em EON, comparada com a uma solução RSA convencional. A principal delas é o aumento significativo de requisições de grandes demandas sendo servidas dinamicamente.

Este trabalho também propõe três soluções cientes da aplicação, em um cenário com duas aplicações distintas concorrendo por recursos. Por serem críticas, o objetivo principal é manter o funcionamento bem sucedido das ressincronizações, sem elevar a taxa de bloqueio de banda para *backups*.

3. Problema das Transferências de Dados em Massa para as aplicações de Backup e Ressincronização ICD

A computação em nuvem tem permitido que mais dados sejam produzidos por várias gamas de aplicações. Assim, os requisitos de alta disponibilidade, redundância e proteção contra falhas são partes fundamentais das garantias de confiabilidade e funcionamento contínuo dos serviços. Operações de missão crítica como *backups* e ressincronização são implementadas para fornecer disponibilidade e proteção contra falhas [Agrawal et al. 2013].

Para tolerar uma maior variedade de falhas, conjuntos de réplicas são implantadas de maneira geograficamente distribuída. Cada CD da rede armazena uma partição de dados, que é replicado separadamente. Um grupo de replicação é um grupo de CDs responsável pela mesma partição. Assim, cada CD pode participar simultaneamente de vários grupos diferentes. O Sistema de Gerenciamento de Dados Distribuídos (SGDD) que lida com esses CDs possue um correto e consistente mapeamento dos grupos de replicação em seus membros e suas respectivas localizações [Sharov et al. 2015].

Em cada um dos grupos de replicação, a comunicação entre seus integrantes ocorre por meio de várias rodadas de trocas de mensagens. Existem variados protocolos que controlam essa comunicação com o objetivo de sempre alcançar um estado comum entre as réplicas de dados. Quando um estado desejável e comum é alcançado, diz-se que as réplicas estão síncronas [Castro and Liskov 2002]. Um CD inativo, que deseja voltar à rede após um período de indisponibilidade e cujo estado encontra-se desatualizado, pode submeter uma mensagem solicitando integrar-se a um dado grupo e assim, acionar o serviço de ressincronização. O grupo de réplicas designado pelo SGDD possui o mesmo estado e um mapeamento das partições a serem replicadas [Agrawal et al. 2013].

A ressincronização ocorre por meio de MBDT originadas de CDs definidos pelo SGDD e recebidas pelo CD desatualizado, ocorrendo dentro de um período de tempo suficiente para a completa atualização do CD solicitante e consumindo recursos na rede. Enquanto *backups* realizam operações individuais de BDT, as ressincronização realizam MBDT de dados relacionados. Ambas são de mesma natureza, pois pertencem à classe de tráfego (*background*) tolerante a atrasos e requererem grandes quantidades de banda [Lu et al. 2015a]. A camada de rede atende várias solicitações de aplicações de diferentes classes de tráfego de forma transparente com roteamento convencional.

As soluções de roteamento ciente da ressincronização ICDs que serão apresentadas, possuem maior complexidade devido a tomada de decisão sobre uma requisição MBDT, feita através da busca de combinações de sub-requisições aptas à transferir partições de dados. Dessa maneira, como qualquer subconjunto de requisições é capaz de realizar uma ressincronização bem sucedida, e atender os requisitos de tolerância a falhas Bizantinas, as demais requisições são redundantes. Para resolver esse problema, além de tentar estabelecer caminhos ópticos com espectro suficiente, essas tentativas são experimentadas para todas os subconjuntos de 3 CDs transmissores que integram o grupo de replicação [Vukolić 2010].

Para prover recursos às BDTs e MBDTs mediante AARSA, a rede foi modelada como um grafo direcionado $G(V, E)$, com V e E representando o conjunto de BW-WXCs e enlaces da fibra, respectivamente. Assume-se que alguns $v \in V$ conectam-se diretamente a CDs, representados como V^{CD}, hábeis a originar e receber transferências de dados. Cada enlace $e \in E$ pode acomodar no máximo, BW *slots* de frequência do espectro.

Requisição BDT. As requisições BDTs são representadas como $r = (s_r, d_r, C_r, dl_r)$ onde s_r e d_r são origem e destino respectivamente, entre os quais uma quantidade de dados C_r é transferida dentro de um prazo dl_r com taxa mínima, $\beta^r_{min} = \frac{C_r}{dl_r}$, ou taxa máxima, β^r_{max}, equivalente a capacidade de banda disponível no canal no momento em que a solicitação é atendida.

Escalonamento de requisições BDTs. Quando uma requisição não pode ser atendida imediatamente, ela é escalonada em uma janela W para novas tentativas do serviço. Na janela, o tempo restante (t^{rem}) para atendimento é equivalente ao tempo atual (t^{now}) subtraído do prazo total (dl_r), ou seja, $t^{rem}_w = dl_r - t^{now}$. A taxa mínima para transferir C_r dentro do período de tempo t^{rem}_w é $\beta^r_{mim_w}$, isto é, $\beta^r_{mim_w} = \frac{C_r}{t^{rem}_w}$, com $\beta^r_{mim} < \beta^r_{mim_w}$.

Requisição MBDT. Uma requisição MBDT é um conjunto de requisições BDT, que é representada como $R = \{r_1, r_2, ..., r_n\}$ onde $dl_{r_i} = dl_{r_2} = ... = dl_{r_n}$. A ressincronização ocorre sobre uma única partição de dados, assim as sub-requisições possuem pequena proximidade de tempo. Para efetiva ressincronização e devido ao limite de tolerância a falhas Bizantinas [Castro and Liskov 2002], três sub-requisições devem ser atendidas. No caso de requisições MBDT com mais do que três requisições de transferência, o algoritmo considera que a aplicação apenas requer o mínimo de três BDTs. Atender a mais do que três sub-requisições leva ao desperdício de recursos. A taxa mínima a ser atribuída para R é definida como β^R_{min}, que é equivalente ao mínimo de banda tal que um número suficiente de $r \in R$ sejam atendidas, e a taxa máxima, β^R_{max}, é equivalente a máxima capacidade de banda disponível no canal no momento do atendimento.

Escalonamento de MBDT. Analogamente ao escalonamento de BDT, quando uma requisição MBDT não pode se servida em uma dada tentativa de busca por recursos, essa requisição é escalonada e espera uma nova oportunidade na janela W. Como o prazo para a transferência diminui, a largura de banda para cada $r \in R$ deve ser atualizada. A taxa mínima para R dentro da janela é $\beta^{R_w}_{min_w} = \sum_1^n \beta^{r_w}_{min_w}$, sendo $\beta^R_{mim} < \beta^{R_w}_{mim_w}$.

As taxas máximas e mínimas para atender BDT podem ser representadas genericamente por $taxa(r)$, e da mesma forma, para atender MBDT, as taxas podem ser generalizadas como $taxa(R)$. Essa notação será utilizada à seguir.

4. Provendo Backups e Ressincronizações

As soluções cientes da aplicação que serão apresentas proveem serviços de *backups* e ressincronização. Serão mostrados os seguintes algoritmos: *(i)* Algoritmo de Roteamento e Alocação de Espectro Ciente da Aplicação Estendido, AARSAE (Algoritmo 1), que atenderá requisições BDT e MBDT; *(ii)* Algoritmo do Máximo de Ressincronizações (MR); e, *(iii)* Algoritmo do Máximo de Ressincronizações e *Backups* (MR+B). Os algoritmos apresentados em (*ii*) e (*iii*) podem invocar as soluções genéricas *ServeBDT* e *ServeMBDT* para a primeira tentativa de servir uma requisição BDT e MBDT, e no caso de requisições serem escalonadas, podem invocar os algoritmos ServeMBDTinW (Algoritmo 2) e ServeRequisicaoInW (Algortimo 3).

4.1. Roteamento e Alocação de Espectro Ciente da Aplicação Estendido (AARSAE)

O AARSAE (Algoritmo 1) atende BDTs e MBDT. Para atender as MBDT o AA-RSA [Sousa et al. 2016] é invocado na linha 2. Para atender as requisições BDT, os

procedimentos das linhas 4-13 são utilizados, começando com a busca de caminhos candidatos pelo $KSP(r, K)$ [Yen 1971]. Em cada caminho verifica-se se a taxa mínima está disponível (linha 6). Em caso positivo, as restrições RSA são testadas (linha 7) para que r seja aceita(linha 8). Caso contrário r é bloqueada (linha 10).

Algoritmo 1 AARSAE(r, R)

```
1: if requisição é R then                                  ▷ Requisição MBDT
2:     AA-RSA(R)
3: else                                                    ▷ Requisição BDT
4:     KSP(r, K)
5:     for k = 1 → K do
6:         if tem β^r_min em k then
7:             if serve restrições RSA then
8:                 Aceita (r, k, β^r_min)
9:             else
10:                Bloqueia r
11:            end if
12:        end if
13:    end for
14: end if
```

4.2. Soluções com Escalonamento

O algoritmo anterior, AARSAE, não faz escalonamento de requisições. Considera-se agora o uso de escalonamento para viabilizar novas reconfigurações de recursos. Ao consumir quase toda a largura de banda disponível, alguns pedidos seriam bloqueados. Com o escalonamento, em vez de bloquear uma requisição devido à falta de recursos, essa requisição é colocada em uma janela W, supervisionada pelo plano de controle da rede, que verifica se o prazo para uma determinada solicitação está se esgotando e é necessário alocar o máximo de banda suficiente para atendê-la. As reconfigurações ocorrem nas chegadas e partidas de requisições, e quando o tempo de uma solicitação na janela atinge o limite.

4.2.1. Soluções Genéricas

Existem dois algoritmos definidos como Soluções Genéricas: *ServeBDT* e ServeMBDT. O ServeBDT é similar aos procedimentos das linhas 4-13 do algoritmo AARSAE (Algoritmo 1), exceto por uma razão: a *taxa* a ser atribuída para atender BDT também é um parâmetro de entrada, $(r, taxa(r))$, e pode ser definida pelo algoritmo que o invocar. O algoritmo ServeMBDT é similar ao AA-RSA [Sousa et al. 2016], exceto por duas diferenças básicas: *(i)* os parâmetros de entrada para ServeMBDT são $(R, taxa(R))$; e *(ii)* a $taxa(R)$ não é atribuída em função de comparações com o diâmetro da rede, conforme o AA-RSA [Sousa et al. 2016], mas sim, é definida pelo algoritmo que o invocar.

4.2.2. Reconfiguração Pós-Escalonamento

Se uma requisição não pode ser servida na primeira tentativa, ela é encaminhada para uma janela de espera W. Em W existem dois procedimentos definidos, o *ServeMBDTinW* (Algoritmo 2) que escalona ressincronizações, e o *ServeRequisicaoInW* (Algoritmo 3), que escalona *backups* e ressincronizações, atendendo as ressincronizações com maior

prioridade. Ambos os algoritmos invocam as rotinas *ServeBDT* e *ServeMBDT* (Soluções Genéricas), passando como parâmetro o tipo de chamada e taxa solicitada por tal chamada.

O algoritmo *ServeMBDTinW* (Algoritmo 2) coloca requisições MBDT na fila $\mathscr{F}^R$ ordenada pelo prazo (linha 1). Para cada elemento nessa fila (linha 3) verifica-se se taxa solicitada é mínima (linhas 4-6) ou máxima (linha 8). A taxa mínima equivale a soma de todas as taxas mínimas de sub-requisições, calculadas a partir do tempo restante para atendimento (linha 6). A taxa máxima equivale a máxima disponibilidade na rede. O algoritmo *ServeMBDT* é requisitado na linha 12. Como a janela contendo as requisições MBDT é monitorada pelo plano de controle, se alguma chamada estiver alcançando o prazo final de atendimento antes de ser definitivamente bloqueada, o algoritmo *ServeMBDT* é requisitado diretamente para esta chamada.

Algoritmo 2 ServeMBDTinW($R, taxa(R)$)

```
 1: 𝓕^R ← R
 2: for R_w ← 1 to |𝓕^R| do
 3:     for r_w ← 1 to |R_w| do
 4:         if taxa(R) = β^R_min then
 5:             t^rem_w = dl_r − t^now
 6:             taxa(r)_w = (C/t^rem_w)
 7:         else
 8:             taxa(R)_w ←Máximo disponível
 9:         end if
10:         taxa(R)_w ← Max(taxa(r)_w)
11:     end for
12:     ServeMBDT(R_w, taxa(R)_w)
13: end for
```

O algoritmo *ServeRequisicaoInW* (Algoritmo 3) trata requisições MBDT e BDT enfileiradas. Para atender as MBDT ele invoca o algoritmo *ServeMBDTinW*, linha 5. Para atender as BDTs da fila, primeiramente verifica-se se não existem MBDTs esperando (linha 7). Em seguida, processa-se a taxa solicitada, que pode ser a mínima, considerando-se o prazo restante (linhas 9-11), ou pode ser a máxima disponível no canal (linha 13). A taxa definida e a requisição BDT em questão são passadas ao algoritmo *ServeBDT* na linha 15. Alertas também são emitidos pelo plano de controle para atender MBDT ou BDT que estejam próximas de encerrar o prazo de atendimento.

4.2.3. Máximo de Ressincronizações (MR)

As soluções prévias possuem os parâmetros genéricos $taxa(r)$ e $taxa(R)$ para que diferentes taxas possam ser solicitadas para atender r e R, sejam taxas mínimas ou máximas. No algoritmo MR (Algoritmo 4), que atende r (linhas 3-4) e R (linhas 6-10), as taxas solicitadas equivalem à taxa máxima disponíveis no canal no momento da solicitação. Assim, um grande número de requisições pode ser admitido, desocupando recursos tão rápido quanto possível. O algoritmo MR faz escalonamento de MBDT, o que significa que após a primeira tentativa de atendimento de uma chamada R (linha 6), se for mal sucedida (linha 7), todas as requisições desse tipo são encaminhadas para uma fila de espera, com a invocação do algoritmo *ServeMBDTinW* (linha 8), e podem ser submetidas a qualquer momento à novas reconfigurações, enquanto o prazo total não tiver transcorrido. Por esse

Algoritmo 3 ServeRequisicaoInW($(r, taxa(r))$,$(R, taxa(R))$)

```
1: 𝓕 ← R ∪ r
2: Ordena 𝓕 priorizando R
3: for f ← 1 to |𝓕| do
4:     if f = R then
5:         ServeMBDTinW(R, taxa(R))
6:     else
7:         if ¬∃R_w(R_w ∈ 𝓕) then
8:             for r_w ← 1 to |𝓕| do
9:                 if taxa(r) = β^r_min then
10:                    t^rem_w = dl − t^now
11:                    taxa(r)_w = (C/t^rem_w)
12:                else
13:                    taxa(r)_w ← Máximo disponível
14:                end if
15:                ServesBDT(r_w, taxa(r)_w)
16:            end for
17:        end if
18:    end if
19: end for
```

motivo, a janela W sempre é verificada. As BDTs não são escalonadas e seu atendimento é feito invocando a rotina *ServeBDT*. Caso a requisição não seja atendida, ela é bloqueada.

Algorithm 4 MR(G, r, R)

```
1: Verifica W
2: for i ← 1 to (Σ(R) + Σ(r)) do
3:     if i = r then
4:         ServeBDT(r, β^r_max)
5:     else
6:         ServeMBDT(R, β^R_max)
7:         if Se R não for atendida then
8:             ServeMBDTinW(R, β^R_max)
9:         end if
10:    end if
11: end for
```

4.2.4. Máximo de Ressincronizações e *Backups* (MR+B)

No algoritmo MR+B (Algoritmo 5), as rotinas *ServeBDT* e *ServeMBDT* são invocadas, cada uma para atender sua respectiva chamada (r e R). Ao tentar atender BDT (linha 4), se a chamada não puder ser atendida (linha 5), ela é imediatamente escalonada e o algoritmo *ServeRequisicaoInW* (linha 6) faz novas tentativas. De maneira similar, as chamadas MBDT tentam ser servidas pelo algoritmo *ServeMBDT* (linha 9), e caso a tentativa falhe (linha 10), essa chamada pode ser escalonada para esperar por outras tentativas de atendimento quando é invocado o algoritmo *ServeRequisicaoInW* (linha 11).

Em todas as oportunidades, as BDTs solicitam a taxa mínima, inclusive quando o algoritmo *ServeRequisicaoInW* é invocado (linha 5). O algoritmo MR+B atribui maior prioridade à MBDT devido a sua maior complexidade e necessidade de banda. A ideia é que a taxa de bloqueio de BDT não cresça e ainda assim, que estas não saturem os recursos disponíveis para que as ressincronizações continuem sendo atendidas.

Algoritmo 5 MR+B(G, r, R)

1: Verifica W
2: **for** $i \leftarrow 1$ to $(\Sigma(r) + \Sigma(R))$ **do**
3: **if** $i = r$ **and** $ServesBDT(r, \beta^r_{mim})$ não atende r **then**
4: $ServeRequisicaoInW(i, taxa(i))$, $taxa(i) = \beta^r_{mim}$
5: **end if**
6: **if** $i = R$ **and** $ServeMBDT(R, \beta^R_{max})$ não atende R **then**
7: $ServeRequisicaoInW(i, taxa(i))$, $taxa(i) = \beta^R_{max}$
8: **end if**
9: **end for**

Complexidade dos Algoritmos

O AARSAE (Algoritmo 1) lida com requisições BDTs e MBDTs. Para atender MBDT, o algoritmo AA-RSA [Sousa et al. 2016] é invocado, e sua complexidade de tempo é $O\left(\left(\frac{n!}{b!(n-b)!}\right) * V^3\right)$. O serviço para BDT apenas requer a busca por caminhos feita pelo KSP e política de atribuição *First-Fit*. Assim, a complexidade de tempo total para esse algoritmo é de $O\left(\left(\left(\frac{n!}{b!(n-b)!}\right) * V^3\right) + K^3V^3\right)$.

Os algoritmos ServeBDT e ServeMBDT, que são Soluções Genéricas (Subseção 4.2.1), lidam com BDT e MBDT, respectivamente. A complexidade de tempo do Algoritmo *ServeBDT* é $O(K^3V^3)$, e a do Algoritmo *ServeMBDT* é $O\left(\left(\frac{n!}{b!(n-b)!}\right) * V^3\right)$.

Os Algoritmos 2 e 3 escalonam requisições e por isso realizam reconfiguração pós-escalonamento (Subseção 4.2.2) . Existe uma fila de requisições e operações são definidas para atualizar o tempo restante para a transmissão, do qual se deduz a taxa. Essas operações são de complexidade de tempo linear. A atualização para requisições na fila, no pior caso pode ser feita para todas as requisições encaminhadas, respectivamente. Assim, a complexidade de tempo do Algoritmo 2 é $O\left(\left(\frac{n!}{b!(n-b)!}\right) * V^3\right)$ e do Algoritmo 3 é $O\left(\log n \left[(K^3V^3) + \left(\left(\frac{n!}{b!(n-b)!}\right) * V^3\right)\right]\right)$.

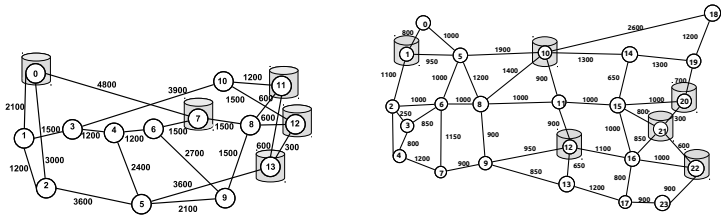
Soluções genéricas e reconfiguração pós-escalonamento são invocadas pelos algoritmos MR e MR+B. Assim, a complexidade de tempo do algoritmo MR é $O\left(K^3V^3 + \left(\left(\frac{n!}{b!(n-b)!}\right) * V^3\right)^2\right)$ e a do algoritmo MR+B é $O\left(\log n \left[(K^3V^3)^2 + \left(\left(\frac{n!}{b!(n-b)!}\right) * V^3\right)^2\right]\right)$.

Embora os algoritmos cientes da aplicação apresentem complexidade de tempo fatorial devido às combinações sobre MBDT, a literatura mostra que o tamanho de b geralmente é de 3 requisições de ressincronizações, visto que no mundo real esse é o tamanho mais comum dos conjuntos de réplicas de centros de dados, por ser desvantajoso,

do ponto de vista do custo capital e operacional, possuir um conglomerado muito grande de recursos que são poucos solicitados [Vukolić 2010].

5. Avaliação de Desempenho

Na avaliação dos algoritmos propostos, que foram implementados no simulador ONS [Costa et al. 2016], eventos dinâmicos de chegadas e partidas de requisições foram simuladas na rede NSFNET (Figura 1(a)) e USA (Figura 1(b)) para comparar o desempenho com o algoritmo RSA [Wan et al. 2012]. Em cada figura abaixo os nós são WXCs, as arestas estão numeradas com as respectivas distâncias e tem-se alguns dos nós destacados por terem conexão direta com CDs que efetuam ou recebem uma transferência de dados.



(a) (b)

Figura 1. Topologias de rede (a) NSFNET e (b) USA.

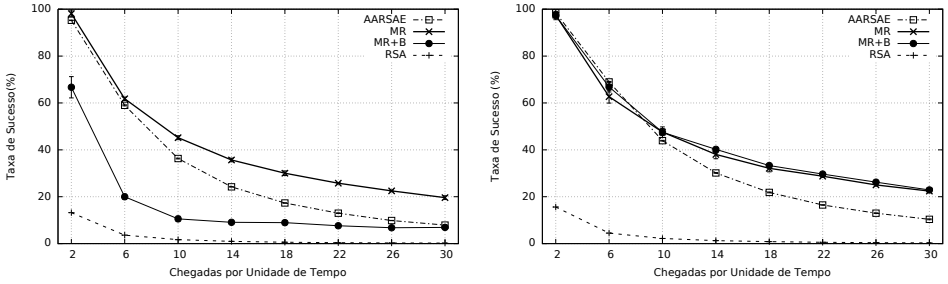
Considerou-se o uso de 15 BVTs em cada nó, sendo cada um com capacidade de transmissão de 8 *slots*. Cada *slot* tem largura de banda de $12.5GHz$ e cada enlace possui 120 *slots* de frequência com espectro total de $1.5THz$. Para banda de guarda assumiu-se dois *slots*. Os algoritmos foram testados usando a modulação *Quadrature Phase Shift Keyed* (QPSK).

Cada simulação foi realizada 5 vezes utilizando o método de replicações independentes. Para os resultados apresentados foram calculados intervalos de confiança com 95% de confiabilidade. Foram geradas 100.000 chamadas por simulação, considerando-se os dois tipos de aplicação em questão, com origens e destinos distribuídos uniformemente dentro do subconjunto de localizações dos CDs. O número de chegadas variou entre 2 e 30 por unidade de tempo [Zhang et al. 2015] com incrementos de 4. As BDTs foram configuradas para transferir $100GB$ e $300GB$ dentro de um prazo de 20 unidades de tempo. Para as MBDTs foram definidas um total de 4 requisições para um mesmo destino, com volume de dados de $100GB$ e $500GB$ para serem transferidos dentro de um prazo de 100 unidades de tempo.

Taxa de Sucesso das Ressincronizações (MBDT)

A taxa de sucesso da ressincronização é calculada como o número de chamadas MBDT aceitas, dividido pelo total de requisições do mesmo tipo. O roteamento convencional não lida com requisições agrupadas (conjunto de requisições relacionadas), assim, precisa servir individualmente e independentemente cada chamada. Já o roteamento ciente da aplicação, lida com conjuntos de requisições relacionadas fazendo combinações em vários subconjuntos com três requisições a partir do conjunto R e selecionando um subconjunto. Isso significa que, em conjuntos com 4 requisições, uma delas será descartada.

A Figura 2(a) mostra os resultados para os algoritmos cientes da aplicação na rede NSFNET. O algoritmo RSA [Wan et al. 2012] tem desempenho inferior aos demais por atender chamadas individualmente e não são direcionadas a atualizar a mesma partição de dados, ao contrário dos algoritmos que são cientes da aplicação. Para atender qualquer tipo de chamada, a taxa mínima é atribuída, tornando os canais ocupados por períodos mais longos. Com o aumento do número de chegadas, as tentativas de satisfazer as solicitações de um conjunto falham por falta de banda disponível, e o esgotamento do prazo também influencia nesse resultado. Sua taxa de sucesso começa em 13.2% para cargas mais baixas, e cai para 0.2% com a saturação dos recursos.



(a) (b)

Figura 2. Taxa de sucesso de ressincronizações nas redes (a) NSFNET e (b) USA.

O algoritmo MR+B serve as ressincronizações com a taxa máxima, resultando em 66% de atendimento com carga mais baixa. Tanto BDT quando MBDT são escalonadas, mas a prioridade de atendimento é dada às MBDTs que, atendidas com taxa máxima, desocupam a banda utilizada mais rapidamente. Muitos BDTs são estabelecidos e permanecem ativos por longos períodos, por receberem taxa mínima, levando os recursos da rede à exaustão, garantindo taxa sucesso de ressincronizações entre 95% (com carga baixa) e $7,9\%$ (com carga alta) entre todas as requisições desse tipo.

O melhor resultado é obtido pelo algoritmo MR que serve BDT e MBDT com taxas máximas e mantém disponibilidade de banda por mais tempo por desocupá-la com rapidez, alcançando de 98% e 19% de sucesso de MBDT. Quando a carga na rede é baixa, quase 25% de sub-requisições são eliminadas ou descartadas das MBDT. Com carga alta, essa taxa cai para 5%. Ambos MR (Algoritmo 4) e MR+B (Algoritmo 5) escalonam requisições MBDT. Graças às várias oportunidades de atendimento oferecidas a cada chamada, em média 43% (para MR) e 50% (para o MR+B) dessas requisições foram atendidas após a primeira tentativa de serviço. Como os recursos se tornam mais escassos, o escalonamento se mostra vantajoso para as requisições com maiores requisitos de banda.

Os resultados na rede USA (Figura 2(b)) sugerem que em uma rede bem conectada, priorizar as MBDT e oferecê-las a taxa máxima resulta em até 98% de ressincronizações bem sucedidas, quando a carga é baixa. Por esta razão, os algoritmos AARSAE ($98,4\%$), MR ($97,4\%$) e MR+B ($97,2\%$) alcançaram bons resultados. O descarte de requisições pelo algoritmo MR é de até $24,6\%$, enquanto o algoritmo MR+B elimina $24,5\%$ e o AARSAE, $24,7\%$. Cerca de 42% das ressincronizações bem sucedidas dos MR e MR+B ocorrem depois que as MBDT são escalonadas.

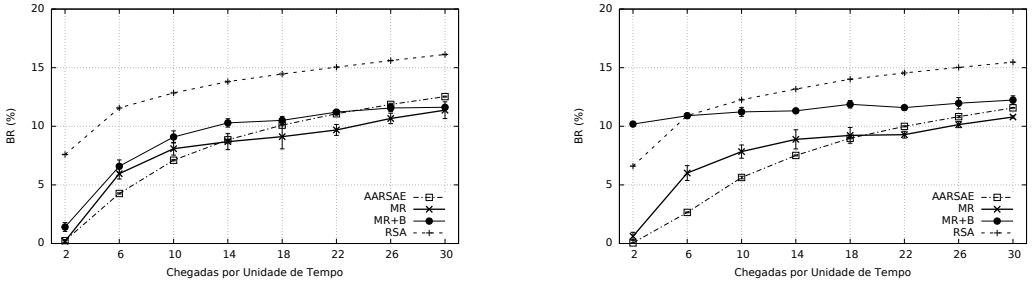
Em geral, as soluções cientes da aplicação podem ser favorecidas quando diferentes

taxas são alocadas e com a combinação de sub-requisições a partir de uma requisição MBDT, que descarta uma solicitação desnecessária de cada MBDT. Com isso economizou-se largura de banda com todas as soluções cientes da aplicação.

Taxa de Bloqueio de Requisições de *Backups* (BR)

A taxa de bloqueio (BR) de *Backups* (BDT) equivale ao número de requisições r bloqueadas, divididas pelo total de requisições desse mesmo tipo. O objetivo das soluções propostas é manter taxa de aceitação das MBDTs em um cenário com mais aplicações executando. No entanto, é desejável que, com o aumento na taxa de aceitação de requisições MBDT, as demais aplicações não sejam penalizadas com bloqueio elevado.

A Figura 3(a) compara a BR de requisições BDTs quando o número de chegadas aumenta de 2 para 30 chegadas de requisições por unidade de tempo na rede NSFNET. O RSA [Wan et al. 2012] possui elevada BR, variando entre $7,5\%$ e 16%, devido às MBDTs que também requisitam serviço e ocupam a banda por mais tempo.



(a) (b)

Figura 3. Probabilidade de bloqueio de BDTs nas redes (a) NSFNET e (b) USA.

O algoritmo MR+B, por outro lado, escalona os dois tipos de requisições existentes, BDTs e MBDTs, o que contribui para uma menor BR, que permanece entre $1,4\%$ e $11,6\%$. Os algoritmos AARSAE (Algoritmo 1) e MR não escalonam requisições BDTs e proveem a mínima e máxima largura de banda disponível, respectivamente, para esses *backups*, mostrando dessa maneira que, quanto mais se pode alocar, menor é a taxa de bloqueio. Adicionalmente, para servir MBDT , o AARSAE define a taxa com base na comparação do tamanho dos caminhos candidatos à metade do diâmetro da rede, passando por grandes variações de disponibilidade de recursos para atender BDTs. O MR sempre escalona MBDTs atribuindo-lhes taxa máxima, contribuindo para reduzidas BR de *backups*.

Na rede USA (Figura 3(b)), os algoritmos que fazem escalonamento tem um padrão diferenciado de comportamento em comparação com as demais topologias. O algoritmo MR+B escalona BDT e MBDT, entretanto, a prioridade atribuída à MBDT em detrimento das BDTs não afetam o seu resultado, visto que em média de 73% de todas as requisições BDTs são aceitas antes de necessitarem do escalonamento. Entretanto, o algoritmo MR+B mantém a taxa de bloqueio entre 10% e 12%.

O algoritmo MR escalona apenas MBDT. Como essas requisições MBDT são sempre atendidas com a taxa máxima, a indisponibilidade de banda acaba impactando no bloqueio de BDTs, com um aumento de 10% até a ocorrência de 18 chegadas de requisições por unidade de tempo, ou seja, quando a carga na rede começa a crescer. O algoritmo AARSAE tem a mais alta variação de crescimento na taxa de bloqueio, que

vai de $0,05\%$ para mais de 11%. Sua política de comparar os caminhos candidatos ao diâmetro da rede para definir a taxa para uma requisição provê bons resultados para as MBDT, mas satura rapidamente os recursos impedindo que as BDTs sejam atendidas.

Em todas as topologias, os algoritmos cientes da aplicação obtiveram menor bloqueio de BDTs em comparação com o algoritmo convencional. Na rede menos conectada (NSFNET), o escalonamento de BDTs reduz a taxa de bloqueio, devido a maior probabilidade de liberação de banda. Na rede mais conectada (USA), o atendimento mais rápido de requisições que demandam muita largura de banda resultou em carga efetiva reduzida na rede. Todavia, essa diminuição na disponibilidade não afetou o serviço de *backups*.

6. Conclusão

Este estudo comparou soluções de roteamento cientes da aplicação e soluções convencionais em EON, executando aplicações BDT e MBDT. Os resultados mostram que, se a solução é ciente da aplicação, o percentual de ressincronizações efetuadas com sucesso pode chegar a 70% se comparada a uma solução convencional. Esse resultado é alcançado sem aumentar a taxa de bloqueio, o que não é possível com o roteamento convencional, cuja taxa de bloqueio foi 6% maior. Além disso, as soluções apresentadas mostram-se mais eficientes em termos de uso de espectro e BVTs em todas as topologias de rede analisadas.

Referências

Agrawal, D., El Abbadi, A., Mahmoud, H., Nawab, F., and Salem, K. (2013). Managing geo-replicated data in multi-datacenters. In Madaan, A., Kikuchi, S., and Bhalla, S., editors, *Databases in Networked Information Systems*, volume 7813 of *Lecture Notes in Computer Science*, pages 23–43. Springer Berlin Heidelberg.

Castro, M. and Liskov, B. (2002). Practical Byzantine fault-tolerance and proactive recovery. *ACM Transactions on Computer Systems*, 20(4):398–461.

Cisco (2016). Cisco vni forecast and methodology, 2015-2020. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html`. Acessed em 10/11/2016.

Costa, L. R., Sousa, L. S., de Oliveira, F. R., da Silva, K. A., Júnior, P. J. S., and Drummond, A. C. (2016). Ons: Simulador de Eventos Discretos para Redes ópticas WDM e EON. In *SBRC 2016 - Salão de Ferramentas*, Salvador, Bahia.

Filer, M., Gaudette, J., Ghobadi, M., Mahajan, R., Issenhuth, T., Klinkers, B., and Cox, J. (2016). Elastic optical networking in the microsoft cloud. *Journal of Optical Communications and Networking*, 8(7):A45–A54.

Laoutaris, N., Sirivianos, M., Yang, X., and Rodriguez, P. (2011). Inter-datacenter bulk transfers with netstitcher. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 74–85. ACM.

Li, Y., Wang, H., Zhang, P., Dong, J., and Cheng, S. (2012). D4d: Inter-datacenter bulk transfers with isp friendliness. In *Cluster Computing (CLUSTER), 2012 IEEE International Conference on*, pages 597–600. IEEE.

Lu, P., Zhang, L., Liu, X., Yao, J., and Zhu, Z. (2015a). Highly efficient data migration and backup for big data applications in elastic optical inter-data-center networks. *IEEE Network*, 29(5):36–42.

Lu, W. and Zhu, Z. (2015). Malleable reservation based bulk-data transfer to recycle spectrum fragments in elastic optical networks. *Lightwave Technology, Journal of*, 33(10):2078–2086.

Lu, W., Zhu, Z., and Mukherjee, B. (2015b). Data-oriented malleable reservation to revitalize spectrum fragments in elastic optical networks. In *Optical Fiber Communications Conference and Exhibition (OFC), 2015*, pages 1–3.

Markowski, M. (2016). Utilization balancing algorithms for dynamic multicast scheduling problem in eon. *International Journal of Electronics and Telecommunications*, 62(4):363–370.

Nandagopal, T. and Puttaswamy, K. P. (2012). Lowering inter-datacenter bandwidth costs via bulk data scheduling. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 244–251. IEEE.

Sadasivarao, A., Naik, D., Liou, C., Syed, S., and Sharma, A. (2016). Demystifying sdn for optical transport networks: Real-world deployments and insights. *IEEE GLOBECOM 2016*.

Sharov, A., Shraer, A., Merchant, A., and Stokely, M. (2015). Automatic reconfiguration of distributed storage. In *Autonomic Computing (ICAC), 2015 IEEE International Conference on*, pages 133–134. IEEE.

Song, F., Huang, D., Zhou, H., and You, I. (2012). Application-aware virtual machine placement in data centers. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pages 191–196. IEEE.

Sousa, L. S., Costa, L., Rodopoulos, F., Drummond, A., and Alchieri, E. (2016). Roteamento e Alocação de Espectro Ciente da Aplicação em Redes Ópticas Elásticas. In *SBRC 2016 - Trilha Principal*, Salvador, Bahia.

Subramaniam, S., Brandt-Pearce, M., Demeester, P., and Saradhi, C. V. (2013). *Cross-layer design in optical networks*. Springer.

Vukolić, M. (2010). The byzantine empire in the intercloud. *ACM SIGACT News*, 41(3):105–111.

Wan, X., Hua, N., and Zheng, X. (2012). Dynamic routing and spectrum assignment in spectrum-flexible transparent optical networks. *Journal of Optical Communications and Networking*, 4(8):603–613.

Yen, J. Y. (1971). Finding the k shortest loopless paths in a network. *management Science*, 17(11):712–716.

Zhang, H., Chen, K., Bai, W., Han, D., Tian, C., Wang, H., Guan, H., and Zhang, M. (2015). Guaranteeing deadlines for inter-datacenter transfers. In *Proceedings of the Tenth European Conference on Computer Systems*, EuroSys '15, pages 20:1–20:14, New York, NY, USA. ACM.

Zinner, T., Jarschel, M., Blenk, A., Wamser, F., and Kellerer, W. (2014). Dynamic application-aware resource management using software-defined networking: Implementation prospects and challenges. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–6. IEEE.

Trilha Principal do SBRC 2017
Sessão Técnica 10
Internet das Coisas

CoAP-CTX: Extensão Sensível ao Contexto para Descoberta de Objetos Inteligentes em Internet das Coisas

Felipe M. Barreto, Windson Viana, Marcio E. F. Maia, Rossana M. de C. Andrade*

Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat)
Mestrado e Doutorado em Ciências da Computação (MDCC)
Universidade Federal do Ceará (UFC)
Fortaleza – CE – Brasil

{felipebarreto, windson, marcio, rossana}@great.ufc.br

Abstract. *In the Internet of Things vision, smart objects are interconnected in order to allow the creation of applications embedded in everyday environments (e.g., house, cars, schools, buildings). The number of smart objects tends to increase in the next years, creating an overload of objects to be controlled and configured by the users. Alternatively, Context-aware discovery services have the potential to minimize this problem by applying rules to determine wich smart objetcs will be discovered at a certain time. This work proposes the CoAP-CTX, an extension of the built-in discovery service present in CoAP protocol that aims to provide support to a selective discovery of smart objects. Aditionally, smart objetcs that are not on the user's interest go into an idle state, optimizing the network and battery usage. Experiments have shown that CoAP-CTX reduces the total number of messages transmitted in the local network, at a cost of a acceptable latency overhead to perform the discovery.*

Resumo. *Em um ambiente de Internet das Coisas, objetos inteligentes estão interligados de modo a permitir a criação de aplicações em lugares da vida cotidiana (por exemplo, casas, carros, escolas, edifícios). O número desses objetos inteligentes tende a aumentar mais ainda nos próximos anos, criando uma sobrecarga de objetos a serem controlados e configurados pelos usuários. Serviços de descoberta sensíveis ao contexto tem o potencial de minimizar este problema aplicando regras para determinar quais objetos inteligentes serão descobertos a cada vez. Este trabalho propõe o CoAP-CTX, uma extensão do serviço de descoberta padrão do protocolo CoAP que tem por objetivo dar suporte a uma descoberta seletiva de objetos inteligentes. Além disso, objetos inteligentes que não são de interesse do usuário entram em um modo de espera, otimizando o uso da rede e da bateria. Experimentos mostraram que o CoAP-CTX reduz o número total de mensagens transmitidas na rede local, a um custo de um aumento aceitável no tempo gasto para realizar a descoberta.*

1. Introdução

Internet das Coisas, do inglês *Internet of Things* (IoT), é um paradigma tecnológico que emerge no cenário já consolidado das redes de comunicações sem fio. A ideia principal desse conceito é a constante e invisível presença, no cotidiano das pessoas, de uma

*Bolsista de produtividade DT-2 do CNPq

enorme variedade de dispositivos computacionais. Alguns desses dispositivos possuem poder computacional, juntamente com a capacidade de comunicação, e são chamados de Objetos Inteligentes (OI). Esses objetos são unicamente endereçáveis, capazes de interagir, trocar dados entre si e ainda cooperar com seus vizinhos para realizarem tarefas em comum [Atzori et al. 2010]. Ao longo dos últimos anos, o volume mundial de OI cresceu rapidamente. A previsão é de que no ano de 2020 será alcançada uma marca de 50 bilhões de OI conectados [Dave 2011] e o número de dispositivos conectados à rede pode chegar à 13,6 por pessoa [Cisco 2016]. Grande parte desses OI já são compatíveis com arquiteturas e protocolos Web (HTTP, REST, entre outros).

As abordagens de descoberta de OI tradicionais tem por objetivo principal descobrir e tornar acessível todos os OI alcançáveis. Entretanto, se o número de OI for muito elevado, uma abordagem mais apropriada seria priorizar, ou recomendar, os OI de maior interesse para o usuário, por exemplo, a partir do contexto capturado por seu *smartphone* (e.g, localização, histórico, entre outros). Além das vantagens diretamente relacionadas ao usuário, com uma descoberta mais seletiva é possível economizar os recursos computacionais desses OI, os quais possuem por natureza limitações de energia e memória [Duarte et al. 2014].

Dispositivos móveis, como o *smartphone*, são uma ótima escolha para serem utilizados como elementos que permitem a interação com OI que estejam presentes ao redor do usuário, porém apresentam limitações com relação a suficiência energética [Ríos et al. 2016]. Dentre as principais operações que consomem energia nos dispositivos móveis, as que mais têm impactado para o aumento desse consumo são as operações de comunicação, como troca de mensagens [Tarkoma et al. 2014]. Dispositivos ainda mais limitados computacionalmente, como é o caso da grande parte dos OI existentes, sofrem do mesmo problema energético, porém de maneira mais intensificada.

A seguir, é apresentado um cenário motivador de descoberta de OI em IoT em que é possível identificar onde a sensibilidade ao contexto pode ser utilizada. Ana é uma pessoa bastante ligada ao uso de tecnologia, costuma acordar cedo para ir ao trabalho, mas não sem antes conferir a previsão do tempo pela TV. Com seu *smartphone*, Ana consegue acessar e ligar a TV, bem como selecionar o canal da previsão. Ela então segue para o trabalho. Usando o *smartphone*, seleciona o noticiário econômico no rádio do carro. Ao chegar no seu destino, Ana prepara a sala de seminários da empresa para uma reunião, novamente utiliza seu *smartphone* para ligar o projetor e as luzes do ambiente.

O cenário apresentado possui três ambientes distintos: a casa, o carro e o local de trabalho. A sensibilidade ao contexto atua em todos eles. Na situação da casa e do carro, embora a tendência seja que o número de dispositivos conectados aumente, pode-se assumir que o volume total ainda será controlado. Nesses casos, a sensibilidade ao contexto pode atuar na melhoria do gerenciamento dos dispositivos, reduzindo o consumo de energia. Por exemplo, os OI que não fossem do interesse de Ana naquele momento (televisão e rádio) poderiam entrar em modo de espera. Já na outra situação, em um ambiente de trabalho, geralmente o número de OI é bem maior, e geralmente são compartilhado por vários usuários (microfones, caixas de som, projetores, etc.). Nesse caso, a vantagem de se utilizar de técnicas de sensibilidade ao contexto na descoberta desses objetos está na interação entre usuário e sistema.

Para a comunicação com os OI, já existem protocolos especializados em dispositivos limitados computacionalmente, como é o caso do *Constrained Application Protocol*[1] (CoAP) e do MQ Telemetry Transport[2] (MQTT). Este trabalho apresenta uma extensão do CoAP, protocolo de troca de mensagens que já possui um serviço de descoberta de OI, de modo a permitir que essa descoberta utilize informações contextuais para selecionar os OI de interesse do usuário. Desse modo, é possível identificar e fazer com que os OI que não sejam de interesse entrem em modo de espera, reduzindo o número de mensagens trocadas na rede.

O restante deste documento está organizado como segue: a Seção 2 apresenta definições e conceitos utilizados neste trabalho. A Seção 3 descreve a extensão do CoAP proposta, bem como um estudo de caso implementado em um ambiente real. A Seção 4 apresenta a avaliação realizada por meio de simulações. A Seção 5 traz discussões sobre trabalhos relacionados com descoberta de OI. Por fim, a Seção 6 apresenta as considerações finais e trabalhos futuros.

2. Fundamentação Teórica

2.1. Sensibilidade ao Contexto

[Dey et al. 2001] definem contexto como qualquer informação que pode ser usada para caracterizar a situação de um elemento que é relevante para a interação entre usuário e sistema, incluindo como elementos, o próprio usuário e o sistema. A definição que será adotada neste trabalho, proposta por [Viana et al. 2011], pode ser vista como uma extensão da anterior, removendo a limitação do contexto sobre a necessidade de interação entre usuário e sistema. Contexto agora passa a ser um conjunto de informações que podem descrever a situação das entidades (e suas relações) envolvidas em uma ação que seja julgada importante para o sistema (e.g., interações, busca de dispositivos, entre outros). A Figura 1 exemplifica visualmente essa definição.

O contexto é definido como a interseção entre duas zonas. A primeira delas, chamada de Zona de Interesse (ZI), representa o conjunto de entidades que o sistema julga importante em um determinado instante de tempo, ou seja, quais informações contextuais o sistema tem interesse em obter em um instante *t*. A segunda zona, chamada de Zona de Observação (ZO), é composta pelas entidades que podem ser acessadas e obtidas nesse mesmo instante de tempo pela infraestrutura de aquisição de contexto. Essa definição de contexto possui duas características importantes. O contexto é tanto dinâmico (as informações fornecidas por cada uma das entidades muda com o tempo) quanto evolutivo (as próprias entidades que compõem o contexto podem mudar) [Duarte et al. 2015]. Isso faz com que a aplicação dessa definição em sistemas móveis, ubíquos e de IoT seja bastante adequada, pois os elementos que compõem esses sistemas são bastante voláteis, o que faz com que as zonas mudem constantemente.

2.2. Descoberta de Objetos Inteligentes

Descoberta de serviços é um aspecto conhecido do desenvolvimento de sistemas distribuídos. [Crasso et al. 2008] definem essa descoberta como um processo de encontrar os serviços adequados para resolver uma determinada tarefa em particular. Esses serviços

[1]CoAP: https://tools.ietf.org/html/rfc7252

[2]MQTT: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

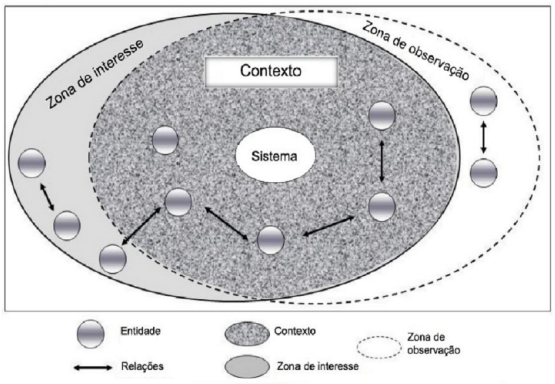


Figura 1. Contexto como sendo a interseção entre as Zonas de Interesse e de Observação [Duarte et al. 2015].

podem ser vistos tanto como abstrações de softwares disponíveis (e.g., um servidor de um jogo) quanto como recursos ou dispositivos computacionais passíveis de serem encontrados (e.g., impressoras em uma rede). *Dynamic Host Configuration Protocol* (DHCP), *Simple Service Discovery Protocol* (SSDP) e *Universal Plug and Play* (UPnP) são exemplos de protocolos que possuem mecanismos de descoberta de serviços. Já para a descoberta de OI em IoT, é preciso levar em consideração requisitos que são específicos para ambientes inteligentes, como os levantados por [Guinard et al. 2010]:

1. **Mínimo *Overhead* no serviço de descoberta:** como a maioria dos OI do mundo real são dispositivos com baixo poder computacional, existe uma necessidade de utilização de paradigmas de descoberta mais leves;
2. **Mínimo esforço no registro de OI:** um OI deve ser capaz de anunciar seus serviços a um servidor de registros através da rede. Esse processo precisa ser realizado sem nenhuma intervenção humana. A quantidade de informações necessárias para esse registro também deve ser bem reduzida;
3. **Suporte à busca contextual e dinâmica de OI:** os algoritmos de busca devem ir além de uma simples busca por palavras-chave. Eles devem levar em consideração dados dinâmicos do contexto do usuário, como localização;
4. **Suporte à alocação de OI sob demanda:** os serviços ofertados pelos OI devem ser ativados sob demanda, evitando assim a má utilização dos recursos.

2.3. CoAP

O CoAP oferece suporte a uma comunicação entre aplicações e OI seguindo o paradigma requisição/resposta. Esse protocolo possui um serviço de descoberta já implementado, baseado no conceito de diretórios de OI. Sua estrutura é baseada no HTTP, facilitando assim a integração com os recursos disponíveis na WEB. Entretanto, diferente do HTTP, o CoAP cumpre alguns requisitos específicos para dispositivos com limitações computacionais, como o baixo *overhead* na troca de mensagens. Clientes utilizam servidores CoAP para acessar uma lista de OI mantida por eles. Cada OI é representado por uma URI, seguindo o formato especificado pelo *Constrained RESTful Environments*[3] (CoRE). A descoberta pode ser realizada com a utilização de filtros, que são parâmetros adicionados a *string* de consulta, enviada ao respectivo diretório de recursos.

[3]CoRE: https://tools.ietf.org/html/rfc6690

O CoRE também define um conjunto de atributos que representam os recursos presentes em um diretório. Os principais atributos são: *Resource Type*, que é o responsável por identificar a função de um determinado recurso (temperatura, luminosidade, impressora, etc.); *Interface Description*, que indica os métodos que podem ser utilizados para a comunicação com esse recurso (*GET, POST*, etc); e *Context Type*, que representa o formato dos dados fornecidos pelo recurso. O CoAP já possui inúmeras implementações disponíveis, para as mais diversas plataformas[4], isso é um bom indicativo que existem inúmeras solução que são compatíveis com essa especificação. Com base nisso, criar soluções que também utilizem o CoAP pode garantir a interoperabilidade com essa gama imensa de dispositivos já ativos. Embora essa interoperabilidade não possa ainda ser estendida para IoT em geral, garantir a compatibilidade com o CoAP é uma estratégia que potencializa a interoperabilidade da proposta deste trabalho.

3. CoAP-CTX

CoAP-CTX CoAP ConTeXtual) é uma extensão do serviço de descoberta do CoAP que visa atender os requisitos de descoberta contextual e alocação de objetos inteligentes sob demanda (Vide Seção 2.2). O CoAP-CTX segue um processo de descoberta sensível ao contexto de objetos inteligentes que pode ser dividido em 8 etapas. (1) Aquisição do contexto do usuário, informações contextuais essas que são a entrada do processo; (2) Inferência dos objetos inteligentes de interesse; (3) Representação e identificação dos OI de interesse; (4) Criação de *strings* de consulta CoAP que realizam uma pré-filtragem; (5) Busca por OI que satisfaçam os filtros da consulta CoAP; (6) Aquisição da lista de objetos inteligentes disponíveis no ambiente e que são de interesse do usuário; (7) Representação dos OI relevantes e disponíveis; (8) Listagem dos OI selecionados, que são a saída do processo. Todas essas etapas podem ser visualizadas na Figura 2.

Na primeira etapa (1), as informações que descrevem o contexto atual do usuário (e.g., localização, situação, atividade, etc.) são obtidas (2) e utilizadas para a identificação de quais OI são de interesse do usuário em um dado momento. A Figura 2 apresenta um interesse que inclui os OI com as seguintes funcionalidades: Controle de acesso (controle das portas); Horário atual; Alarme sonoro; Dispositivo móvel; Impressora. (3) Cada OI é representado por meio de uma única *String* que agrega informações contextuais e que pode ser utilizada para identificação desse dispositivo. Como os OI utilizam o CoAP como protocolo de comunicação e interação, pode-se utilizar mecanismos já disponíveis nesse protocolo, como as *strings* de consulta CoAP, geradas na etapa (4). Nesse caso, um exemplo de consulta ao servidor CoAP seria: *GET /.well-known/core?rt="access-control time alarm mobile-device printer"*. A referência aos OI que se registraram no servidor CoAP (5) e que atenderam às restrições expressas na consulta realizada é retornada ao serviço de descoberta (6). Antes de gerar a lista final de objetos relevantes e disponíveis (7), é preciso verificar as informações dinâmicas do contexto do usuário e dos OI, como localização. Por fim, essa lista final é apresentada ao usuário (8), que por sua vez poderá interagir com esses OI e realizar operações do cotidiano mais facilmente.

A solução proposta foi desenvolvida para atuar em ambientes inteligentes que possuam três tipos de elementos básicos: um *smartphone* Android, um ou mais servidores CoAP e um conjunto de objetos inteligentes. O *smartphone* funciona como o elemento

[4]http://coap.technology/impls.html

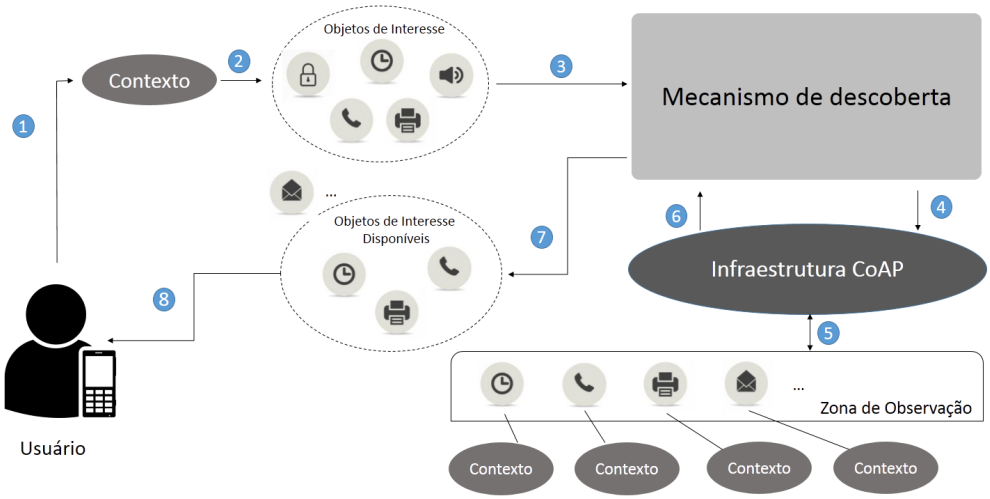


Figura 2. Visão geral do processo de descoberta de objetos inteligentes

coordenador de todo o processo de descoberta, gerenciando assim todos os OI que são de interesse do usuário. A Figura 3 apresenta a arquitetura do CoAP-CTX.

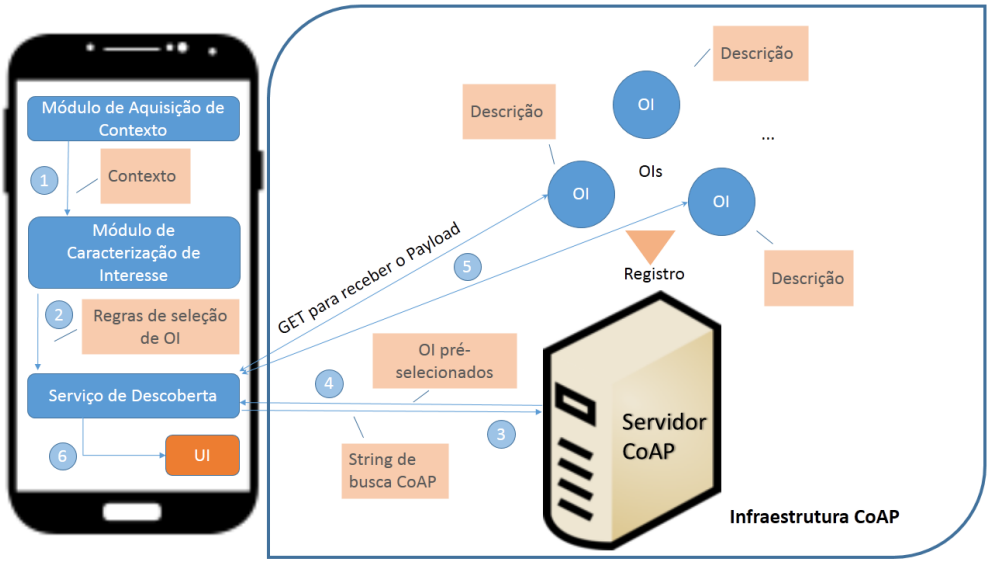


Figura 3. Arquitetura do CoAP-CTX

A primeira tarefa do sistema é obter o contexto (1) por meio do Módulo de Aquisição de Contexto. Esse módulo é responsável pela detecção da situação (e.g., usuário acordou) e fornecimento das informações contextuais suficientes para caracterizar seu interesse naquele momento. Com base nessas informações contextuais, o Módulo de Caracterização de Interesse constrói uma forma simples de representação (2), uma lista de *Strings* contendo características separadas por ”.”, de forma hierárquica. Por exemplo, se o usuário tem interesse em acessar a TV do quarto, esse interesse é representado por *control.ambient.tv* e *context.ambient.quarto*. Esse módulo foi desenvolvido de modo a permitir futuras extensões, com abordagens de caracterização do interesse mais complexas, como *machine learning*.

Com essa representação do interesse, o Serviço de Descoberta pode realizar um mapeamento entre interesse e *String* de consulta CoAP (3). Cada campo das *Strings* de interesse é mapeado para campos que definem e descrevem os objetos inteligentes que seguem a especificação CoRE. Os campos *control e ambient* dizem que o objeto deve ser capaz de atuar no ambiente, o que pode ser representado por um filtro no campo *Interface-Description* do OI, limitando seu valor para apenas *actuator*. Já o campo *tv* da *String* de interesse, representa o tipo, ou classe, do objeto inteligente, e pode ser mapeado para o campo *Resource-Type*, aceitando apenas OI que possuam esse atributo igual a *tv*. Com isso é possível gerar a requisição para o Servidor CoAP em busca de todas as TVs que podem ser controladas pelo usuário.

O Servidor CoAP mantém o registro de todos os OI disponíveis no ambiente, isso inclui até os que não fazem parte do interesse do usuário naquele momento. Ao receber a requisição feita em busca dos OI, o servidor aplica os filtros sobre os campos *Resource-Type* e *Interface-Description* e retorna a lista de OI (4) que atendem à requisição. Pode ocorrer de vários OI serem retornados, por exemplo as TVs do quarto e da sala. Não é possível realizar uma consulta para identificar, ainda no Servidor CoAP, qual é a televisão do quarto especificamente. Isso porque a informação contextual do OI, que é dinâmica, não fica disponível no serviço de descoberta do Servidor CoAP. Para ter acesso a essas informações, clientes CoAP devem realizar uma operação de GET (5) nesses OI e receber seu contexto no *Payload* da mensagem.

O Serviço de Descoberta, ao receber a lista com os OI pré-selecionados pelo servidor, gera uma requisição GET para cada um deles. Com base nas informações contidas no *Payload* da mensagem, é possível aplicar as regras de seleção que envolvem características dinâmicas, como localização. A referência dos OI selecionados é por fim repassada para o gerenciador da interface gráfica da aplicação (6), que é responsável por exibir na tela do *smartphone* os OI selecionados, além de permitir que o usuário os acesse. Todos os outros OI que estão registrados no servidor CoAP são postos em modo de espera, até que o contexto mude e uma nova descoberta aconteça.

3.1. Prova de Conceito

Uma primeira prova de conceito (PoC) foi concebida e implementada como objetivo de ilustrar uma descoberta seletiva de OI, com base no contexto. Para essa PoC, foram implementadas uma aplicação Android, um servidor CoAP baseado na implementação jCoAP[5] e quatro OI por meio da plataforma Arduino (duas TVs, um Ar Condicionado, e um sensor de luminosidade). O contexto utilizado foi o da chegada do usuário na sua casa. A localização, obtida pelo GPS do *smartphone*, detecta o evento da chegada, enquanto que as informações de histórico sugerem que o usuário tem interesse nos OI que controlam o estado da casa, ou seja, atuadores. O filtro aplicado para esse contexto foi no campo *Interface-Description*, selecionando apenas os atuadores, no caso, três dentre os quatro OI. A aplicação Android lista os OI que foram selecionados após a aplicação das regras de seleção (OI colocalizados com o usuário e que funcionam como atuadores), além de permitir que o usuário acesse cada um dos OI selecionados. A Figura 4 apresenta *prints* da tela da aplicação executando. Durante o carregamento da aplicação (a), são realizadas as consultas ao servidor CoAP e aplicadas as regras de seleção. Em (b), são listados os OI

[5]http://www.ws4d.org/ws4d-jcoap/

que satisfizeram as regras de seleção contextual. Já as telas (c) e (d) mostram a interface de controle da TV e do Ar Condicionado (AC), respectivamente.

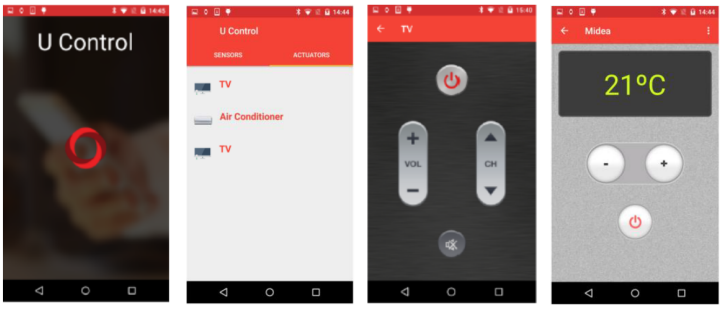


Figura 4. Aplicação Android desenvolvida como prova de conceito.

Para a implementação dos OI, foram utilizados três Arduinos Mega e um Arduino Yún, com interfaces de comunicação *Bluetooth* e *WiFi*, respectivamente. A Figura 5 apresenta os hardwares utilizados para a PoC. Em (a) está o Arduino Yún com o sensor de luminosidade. Em (b) os componentes para controlar o Ar Condicionado. Em (c) são exibidos os componentes utilizados para controlar as TVs (replicação do hardware). O controle das TVs e do Ar Condicionado foi feito utilizando infravermelho (IR), com os códigos IR de cada comando obtidos por engenharia reversa.

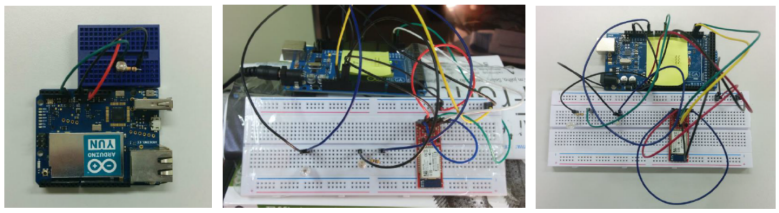


Figura 5. Hardware utilizado na prova de conceito.

4. Avaliação

Esta seção descreve os experimentos realizados para avaliação do CoAP-CTX em três diferentes ambientes inteligentes: casa, carro e prédio.

4.1. Objetivo da Avaliação

A prova de conceito (Vide Seção 3.1) mostrou que o CoAP-CTX pode ser utilizado para reduzir o número de OI apresentados ao usuário, diminuindo a sobrecarga visual que um grande número de OI poderiam causar. Além disso, espera-se que a proposta diminua o número total de mensagens trocadas na rede, no caso em que OI possam entrar em modo de espera. Como o número de mensagens trocadas é proporcional ao consumo de energia, reduzir o primero também tem por objetivo reduzir o consumo energético da rede como um todo. Além disso, como está sendo adicionado uma camada de complexidade acima do serviço de descoberta padrão do CoAP, um aumento do tempo de descoberta

final também é previsto. Logo, o objetivo da avaliação é verificar o comportamento do CoAP-CTX com relação ao número de mensagens trocadas na rede e o tempo gasto para realizar a descoberta de OI.

4.2. Materiais e Métodos

Para essa avaliação, foi utilizado o simulador Cooja[6], que é voltado especificamente para redes de sensores sem fio. O Cooja foi executado por meio de uma máquina virtual Linux, disponibilizada pelos próprios desenvolvedores do simulador. Todos os componentes foram implementados e simulados para a plataforma de *hardware Tmote Sky*[7]. Essa plataforma possui um baixo tempo de transição para sair do modo de espera, além disso, utiliza o módulo CC2420[8] para comunicação sem fio, que é compatível com o padrão IEEE 802.15.4. O Cliente CoAP realiza a descoberta a cada 10 segundos, enquanto os OI que não estão em modo de espera enviam seu estado a cada 200*ms* para o servidor CoAP.

4.3. Procedimento

Os resultados dos experimentos dependem diretamente do contexto do usuário e dos OI, podendo sofrer grandes variações para contextos diferentes. Como forma de contornar esse problema, foram analisados três casos específicos para cada ambiente: o melhor e o pior caso, e um intermediário. Consideramos como melhor caso quando apenas um OI é pré-selecionado e não há necessidade de consulta por informações dinâmicas no *payload* da mensagem desse OI específico. Já o pior caso ocorre quando todos os OI são pré-selecionados e há necessidade da consulta individual, resultando em uma seleção final de todos os OI. O caso intermediário é analisado separadamente para cada cenário simulado, utilizando valores empíricos para a definição do número de OI. Foi então analisada a performance da descoberta usando diretamente o CoAP (sem contexto), e depois usando o CoAP-CTX (com contexto).

Cada ambiente é composto por um cliente CoAP, que representa o *smartphone*, um ou mais servidores CoAP, e um conjunto de OI que se registram nesses servidores. A Figura 6 apresenta a topologia de cada rede gerada. Para cada caso de cada ambiente foram realizadas 10 simulações de 15 minutos, os resultados são apresentados na forma de *box plots*, permitindo a visualização dos valores médios, variância e limites de *outliers*. Foi utilizado apenas um cliente CoAP em cada simulação porque esse cliente representa um único usuário interagindo com o ambiente inteligente em questão. Múltiplos usuários simultâneos trazem novos requisitos que estão fora do escopo deste trabalho, como controle de acesso.

4.4. Casa Inteligente

Casas inteligentes geralmente possuem um número moderado de OI, para a simulação foram utilizados 20 OI, 1 Servidor CoAP e 1 Cliente CoAP. Para o caso intermediário, supôs-se que o interesse do usuário é mais genérico, fazendo assim com que a descoberta retorne um grande número de OI. Nesse caso, 15 OI são pré-selecionados pelas *Strings* de consulta CoAP, desses, 5 são descartados após a aplicação das regras de seleção com informações dinâmicas, totalizando 10 OI ativos e 10 em modo de espera. As Figuras 7

[6]http://anrg.usc.edu/contiki/index.php/Cooja_Simulator

[7]http://wirelesssensornetworks.weebly.com/1/post/2013/08/tmote-sky.html

[8]http://www.ti.com/product/CC2420

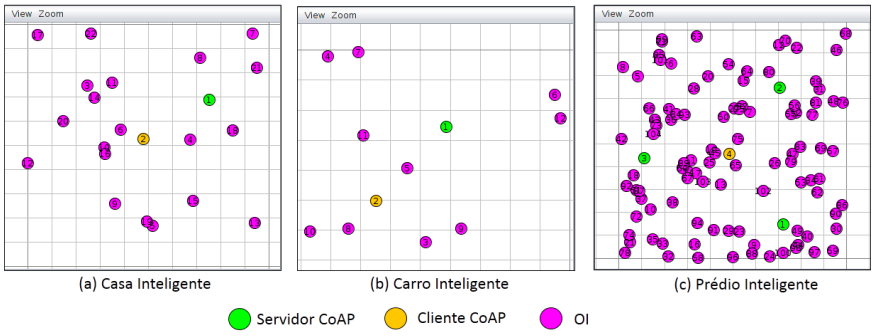


Figura 6. Redes criadas com o Cooja para avaliar os cenários propostos

(a) e (b) mostram os resultados encontrados para esse ambiente, onde é possível observar que uma redução de cerca de 50% do número de OI descobertos gerou uma redução de aproximadamente 41% no número total de mensagens. Já o tempo de descoberta aumentou em 100*ms*, também no caso intermediário, com relação ao CoAP padrão.

4.5. Carro Inteligente

Carros possuem um número de OI mais reduzido, por isso foram utilizados 10 OI, 1 Servidor CoAP e 1 Cliente CoAP. O cenário de utilização dos OI do carro também são mais específicos. Enquanto dirige, o usuário tende a ter mais interesse em OI relacionados a navegação, já os OI de multimídia ganham prioridade quando o carro está parado. Com base nisso, definiu-se o caso intermediário como sendo 4 OI pré-selecionados pelo servidor CoAP, dentre esses 3 se tornam disponíveis ao usuário, totalizando 7 OI em modo de espera. Os resultados, apresentados nas Figuras 7 (c) e (d), mostram que, para o caso intermediário, o CoAP-CTX obteve uma redução de cerca de 57% no número de mensagens, com um aumento no tempo de descoberta médio de apenas 63*ms*.

4.6. Prédio Inteligente

Um prédio inteligente é uma estrutura física que possui uma elevada densidade de OI disponíveis, sendo esses de propósito mais geral, para atender vários usuários. Para esse cenário, utilizou-se um total de 100 OI, 3 Servidores CoAP e um Cliente CoAP. Nesse cenário, informações de contexto dinâmicas, como localização *indoor*, são fundamentais para limitar os resultados da descoberta. Para o caso intermediário, considerou-se que 70 OI seriam pré-selecionados, mas que apenas 15 seriam de interesse do usuário após a obtenção das informações contextuais dinâmicas. As Figuras 7 (e) e (f) mostram os resultados obtidos. Houve uma drástica redução no número de mensagens trocadas na rede quando utilizada a solução proposta, cerca de 80%, para o caso intermediário. Entretanto, para esse caso, o tempo de descoberta médio ficou acima de 1,5 segundos.

4.7. Discussão

Os experimentos mostraram que, em geral, quando comparado ao serviço de descoberta padrão do CoAP, o CoAP-CTX apresenta uma redução no número total de mensagens trocadas na rede, em troca de um aumento no tempo necessário para a realização da descoberta. Porém, esse comportamento se modifica quando as informações contextuais levam

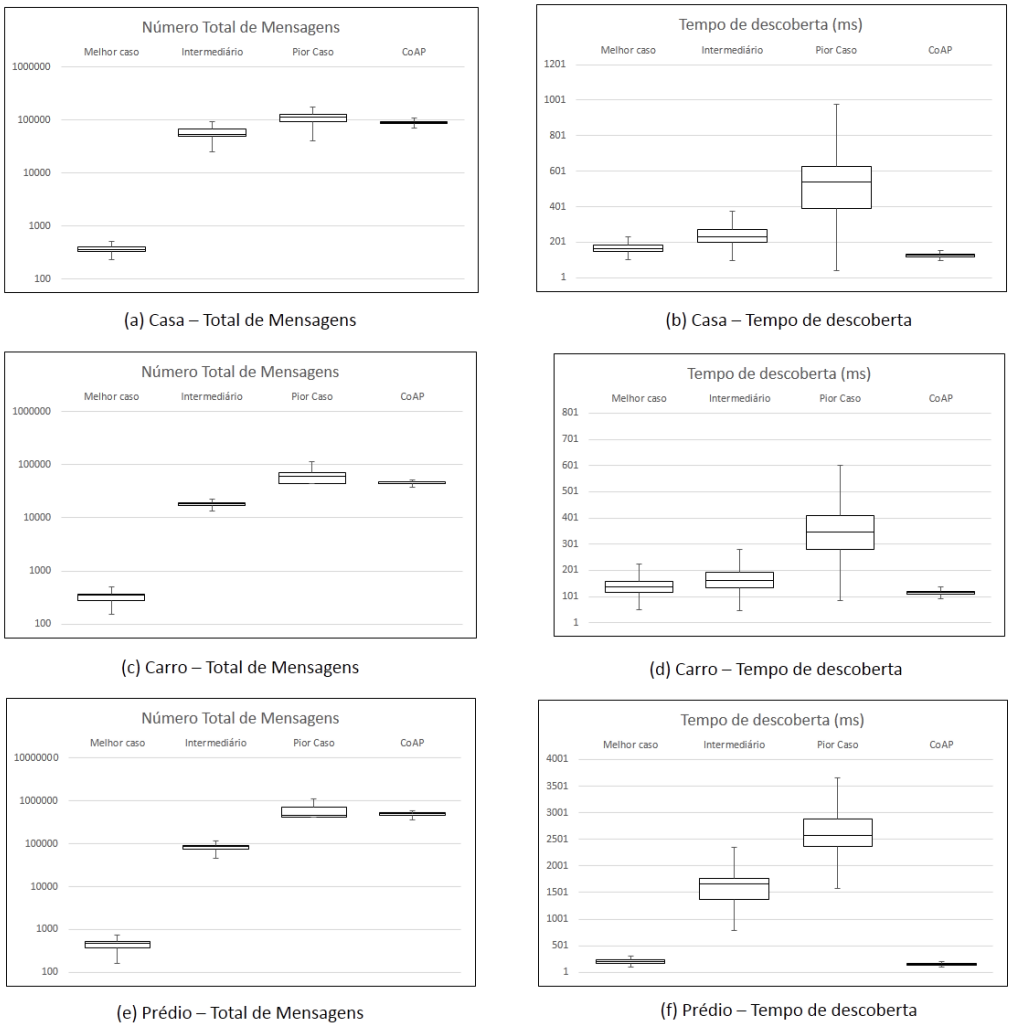


Figura 7. Resultados obtidos para os três ambientes inteligentes: casa, carro e prédio

a um cenário próximo do pior caso, onde o número de mensagens tende a se aproximar dos valores encontrados para o CoAP padrão, podendo até aumentar um pouco. Esse aumento será pequeno pois é devido ao envio de requisições para obtenção de informações contextuais dinâmicas, que representa um percentual bem menor quando comparado a mensagens de atualização dos OI.

Notou-se que o número total de mensagens depende fortemente do número final de OI em modo de espera, enquanto que o tempo de descoberta é afetado pela quantidade de OI com informações dinâmicas a serem obtidas para a aplicação das regras de seleção. Portanto, conclui-se que em cenários onde os OI tem uma função mais especializada, ou seja, em que seja possível pré-selecionar um conjunto próximo do final de OI de interesse, o aumento no tempo de descoberta é baixo. Já em em ambientes genéricos, como prédios inteligentes, onde a maior parte da seleção ocorre com base nas informações contextuais dinâmicas, o tempo de descoberta aumenta consideravelmente. Uma solução para esse aumento seria a migração das regras contextuais dinâmicas para o próprio servidor CoAP,

pois assim todo o processo de descoberta seria realizado com uma única requisição ao servidor CoAP. Entretanto, essa migração faria com que a solução perdesse parte da interoperabilidade que o CoAP proporciona, pois ambientes que já possuem servidores CoAP tradicionais não seriam compatíveis com a aplicação de regras dinâmicas.

5. Trabalhos Relacionados

Nesta seção são discutidos os trabalhos relacionados ao tema de descoberta de OI em um cenário de IoT. Foi dada preferência a trabalhos que propõem algum tipo de serviço ou infraestrutura de descoberta de OI.

[Liu et al. 2013] propõem uma arquitetura de descoberta distribuída de dispositivos focada em IoT. Cada dispositivo é considerado um nó de uma rede P2P e é capaz de realizar o processo de registro, bem como ajudar no serviço de descoberta. A identificação de cada dispositivo é feita utilizando URIs que seguem o modelo CoAP. Por trazerem uma abordagem totalmente distribuída, os autores eliminam os possíveis problemas de um elemento centralizador causados pela mobilidade e volatilidade, intrínsecas ao cenário de IoT. Por outro lado, como cada dispositivo deve ser capaz de realizar o registro, isso pode ser inviável para dispositivos estritamente limitados computacionalmente. Já o Digcovery é uma proposta de descoberta global de OI que usa uma infraestrutura centralizada na qual OI podem se registrar [Jara et al. 2013]. Para o acesso a essa infraestrutura, foi desenvolvido um serviço para dispositivos móveis, que permite a descoberta e o acesso aos objetos inteligentes. Características contextuais de geo-localização são levadas em consideração durante a fase de descoberta, coordenada pelo *smartphone* do usuário.

[Cirani et al. 2014] apresentam uma arquitetura auto-configurável e escalável de descoberta de serviços. No trabalho, é proposta uma topologia P2P com a utilização de diversos *gateways*. Eles são responsáveis por manter a lista de OI presentes na rede. Esses *gateways* são baseados nos servidores CoAP cujos clientes podem realizar uma operação GET para a URI */.well-know/core* a fim de receber a lista de dispositivos disponíveis. DiscoWoT é um serviço de descoberta de OI proposto por [Mayer and Guinard 2011], e que é possível definir estratégias de descoberta em tempo de execução por meio de interfaces RESTful[9]. Esse serviço de descoberta se utiliza de *Microformats* e *Microdata* em conjunto com outras tecnologias WEB para representar semanticamente os objetos inteligentes. A implementação dessa representação é feita utilizando JSON, para garantir a interoperabilidade. [Ishaq et al. 2012] propõem um mecanismo de descoberta baseado em CoAP e DNS. Traduções entre CoAP e HTTP são disponibilizadas, permitindo a descoberta de qualquer objeto inteligente compatível com o padrão *IPv6*.

A Tabela 1 apresenta um comparativo entre esses trabalhos relacionados. Os critérios utilizados na classificação foram escolhidos com base nos requisitos da descoberta de OI. Nota-se que a sensibilidade ao contexto ainda não é amplamente utilizada nas soluções de descoberta de OI para IoT. Apenas algumas das soluções utilizam elementos contextuais e nenhuma combinou o contexto do usuário e do OI em um único serviço. Nota-se também uma certa tendência de utilização do CoAP como base para a descoberta, o que pode ser justificado tanto pela interoperabilidade quanto pelo fato desse protocolo já possuir um serviço de descoberta, que pode ser reutilizado e estendido.

[9]RESTful *web services* são aqueles que seguem o modelo arquitetural REST.

Tabela 1. Comparativo entre os trabalhos relacionados.

Trabalhos Relacionados	CoAP	Elementos Contextuais	Tipo de Contexto
Liu et al.	✓	-	-
Digcovery	-	✓	Usuário
Cirani et al.	✓	-	-
DiscoWoT	-	✓	Objeto Inteligente
Ishaq et al.	✓	-	-
CoAP-CTX	✓	✓	Usuário + OI

6. Considerações Finais

Este trabalho apresenta uma proposta de extensão do CoAP para a descoberta sensível ao contexto de objetos inteligentes. São utilizadas informações contextuais do usuário e dos OI para realizar uma seleção mais direcionada, bem como reduzir o número de mensagens trocadas pelos dispositivos, otimizando assim o consumo de energia. A criação de uma prova de conceito mostrou que a solução é factível e que pode ser utilizada em um cenário real. A avaliação da solução por meio de um simulador garante que, para cenários próximos aos simulados, a utilização do CoAP-CTX reduz o número total de mensagens na rede, reduzindo assim o consumo de energia em geral. Além disso, o aumento do tempo de descoberta é aceitável em cenários onde o interesse do usuário não mude tão rapidamente, característica essa que é comumente atingida.

Como próximos passos desta pesquisa, notou-se a necessidade de tornar a solução mais genérica, permitindo que desenvolvedores estendam os mecanismos de aquisição de contexto e regras de seleção. Logo, o objetivo futuro é transformar o CoAP-CTX em um *framework* para descoberta de OI. Além disso, espera-se realizar mais avaliações, sobretudo para analisar se a lista final de OI descobertos condiz com o real interesse do usuário. A solução atual não oferece suporte a múltiplos usuários, sendo necessária a implementação de uma política de controle de acesso para permitir vários usuários interagindo com os OI ao mesmo tempo. Por fim, para reduzir o tempo de descoberta em ambientes com muitos OI genéricos, pode ser implementado um mecanismo de verificação do servidor CoAP. Logo, as regras de contexto dinâmico podem ser aplicadas por servidores com suporte a essa funcionalidade, ou, caso contrário, executadas no próprio *smartphone*.

Agradecimentos

Gostaríamos de agradecer à CAPES pela concessão da bolsa de apoio à Pós-graduação, que permitiu a realização da pesquisa de mestrado na qual este trabalho foi originado.

Referências

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Comput. Netw.*, 54(15):2787–2805.

Cirani, S., Davoli, L., Ferrari, G., Léone, R., Medagliani, P., Picone, M., and Veltri, L. (2014). A scalable and self-configuring architecture for service discovery in the internet of things. *IEEE Internet of Things Journal*, 1(5):508–521.

Cisco (2016). Cisco global cloud index: Forecast and methodology, 2015-2020. *CISCO white paper*.

Crasso, M., Zunino, A., and Campo, M. (2008). Easy web service discovery: A query-by-example approach. *Science of Computer Programming*, 71(2):144 – 164.

Dave, E. (2011). The internet of things: how the next evolution of the internet is changing everything. *CISCO white paper*.

Dey, A. K., Abowd, G. D., and Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Hum.-Comput. Interact.*, 16(2):97–166.

Duarte, P. A., Gomes, F. A., Barreto, F. M., Viana, W., and Trinta, F. M. (2014). Loccamconfigurator: Uma ferramenta para modelagem visual de configurações de um middleware de suporte à aplicações conscientes de contexto. In *Proceedings of the 20st Brazilian Symposium on Multimedia and the Web*, WebMedia '14.

Duarte, P. A., Silva, L. F. M., Gomes, F. A., Viana, W., and Trinta, F. M. (2015). Dynamic deployment for context-aware multimedia environments. In *Proceedings of the 21st Brazilian Symposium on Multimedia and the Web*, WebMedia '15, pages 197–204, New York, NY, USA. ACM.

Guinard, D., Trifa, V., Karnouskos, S., Spiess, P., and Savio, D. (2010). Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services. *IEEE Transactions on Services Computing*, 3(3):223–235.

Ishaq, I., Hoebeke, J., Rossey, J., Poorter, E. D., Moerman, I., and Demeester, P. (2012). Facilitating sensor deployment, discovery and resource access using embedded web services. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pages 717–724.

Jara, A. J., Lopez, P., Fernandez, D., Castillo, J. F., Zamora, M. A., and Skarmeta, A. F. (2013). Mobile digcovery: A global service discovery for the internet of things. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 1325–1330.

Liu, M., Leppänen, T., Harjula, E., Ou, Z., Ylianttila, M., and Ojala, T. (2013). Distributed resource discovery in the machine-to-machine applications. In *2013 IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems*, pages 411–412.

Mayer, S. and Guinard, D. (2011). An extensible discovery service for smart things. In *Proceedings of the Second International Workshop on Web of Things*, WoT '11, pages 7:1–7:6, New York, NY, USA. ACM.

Ríos, L., Endler, M., and Colcher, S. (2016). An energy-aware iot gateway, with continuous processing of sensor data. XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC2016, Salvador, Brasil.

Tarkoma, S., Siekknen, M., Lagerspetz, E., and Xiao, Y. (2014). Smartphone energy consumption: modeling and optimization.

Viana, W., Miron, A. D., Moisuc, B., Gensel, J., Villanova-Oliver, M., and Martin, H. (2011). Towards the semantic and context-aware management of mobile multimedia. *Multimedia Tools Appl.*, 53(2):391–429.

Pingo d'água: ICMP para Internet das Coisas Aquáticas

Francisco H. M. B. Lima[1], Luiz F. M. Vieira[1], Marcos A. M. Vieira[1], Alex B. Vieira[2], José Augusto M. Nacif[3]

[1]Departamento de Ciência da Computação - Universidade Federal de Minas Gerais (UFMG)

[2]Departamento de Ciência da Computação - Universidade Federal de Juiz de Fora (UFJF)

[3]Universidade Federal de Viçosa (UFV) - Campus UFV Florestal

{francisco.lima, lfvieira, mmvieira}@dcc.ufmg.br

alex.borges@ufjf.edu.br, jnacif@ufv.br

Abstract. *High latency, low transmission rate and the presence of limited computional devices are characteristics present in scenarios of underwater communications. In this paper we present the Water Ping application, which implements the ICMP (Internet Control Message Protocol) protocol with header compression, to work in this scenario. The proposed protocol allows underwater devices to receive ping messages, in this way participating in the Internet and forming the Internet of Underwater Things (IoUT). Results show that underwater network devices can now be remotely monitored by the Internet, and also significative gains in the compression, consuming 93.75% less energy.*

Resumo. *Alta latência, baixa taxa de transmissão e presença de dispositivos computacionalmente limitados são características presentes no cenário da comunicação em ambientes aquáticos. Neste trabalho apresentamos a aplicação Pingo d'água, que implementa o protocolo ICMP (Internet Control Message Protocol) com compressão de cabeçalhos, a fim de atender o cenário. O protocolo proposto permite que dispositivos aquáticos possam receber mensagens ping, dessa forma participando da Internet, e formando a Internet das Coisas Aquáticas (IoUT). Resultados mostram que dispositivos de redes de sensores aquáticas podem ser monitorados remotamente pela Internet e o ganho na compressão é significativo, consumindo 93,75% menos energia.*

1. Introdução

Redes sem fio aquáticas, caracterizadas por terem em sua totalidade ou maior parte dispositivos computacionalmente limitados, tem se tornado foco crescente de pesquisas. Há inúmeras aplicações que podem se beneficiar de pesquisas nesta área, como por exemplo, monitoração de recursos hídricos, monitoração de oleodutos, bacias hidrográficas, reservatórios de hidrelétricas e plataformas de petróleo [Vieira et al. 2010a]. Parte do crescente interesse em redes de sensores aquáticas deve-se ao desenvolvimento de dispositivos capazes de se comunicarem entre si abaixo da superfície aquática, conhecidos como dispositivos aquáticos e nós sensores [Viana et al. 2015].

Apesar da existência de projetos concretos de redes sem fio aquáticas, observa-se uma lacuna quanto à interoperabilidade dessas redes com as redes tradicionais que operam

na Internet. De fato, soluções de comunicação, roteamento e até mesmo de aquisição e análise de dados são criadas fortemente acopladas a um contexto ou tecnologia. Assim, mudanças são necessárias para que esse cenário se torne ubíquo e, análogo à Internet das Coisas, se forme uma Internet das Coisas Aquáticas (IdCA ou IoUT - *Internet of Underwater Things*).

Dada a baixa capacidade computacional dos dispositivos envolvidos e o desafiante canal de comunicação (geralmente acústico), redes sem fio no ambiente aquático não suportam tráfego semelhante ao das redes tradicionais, que operam sobre a arquitetura TCP/IP. Mesmo redes sem fio tradicionais não experimentam características semelhantes a esse meio, como pequena largura de banda, limitação da velocidade do som na água –que representa uma latência cinco ordens de magnitude maior que comunicações na velocidade da luz– e uma notória maior taxa de bits errados [Vieira et al. 2010a]. Além disso, os dispositivos que compõem redes aquáticas apresentam fortes limitações quanto a energia, em especial, por conta de estarem localizados em locais de difícil acesso e manutenção [Coutinho et al. 2016d]. Em suma, nesse cenário faz-se necessária uma comunicação composta por mensagens que sejam curtas e que não ocorram com muita frequência [Sun and Melodia 2013].

Assim, neste trabalho apresentamos uma proposta de compressão para mensagens do protocolo ICMP [Postel 1981] e ICMPv6 [Conta and Gupta 2006] que servirá como bloco base de construção de um arcabouço de IdCA. Note que, a versão do IPv6 para redes sem fio pessoais e de baixo consumo (*6LoWPAN*) [Kushalnagar et al. 2007] apresenta solução para um problema semelhante que ocorre em redes sem fio terrestres e assim, serve como inspiração para o problema tratado em redes sem fio aquáticas. Por fim, as alterações na pilha de protocolos proposta permitirá a monitoração ubíqua dos nós sensores aquáticos, exemplificadas neste trabalho –mas não restrito– pela monitoração realizada por *ping*.

Nós avaliamos as alterações propostas através de testes em um sistema realista, composto por um dispositivo aquático, um dispositivo intermediário (*gateway*) e um computador. Um outro computador remoto foi utilizado para executar o comando Ping, do protocolo ICMP. Assim uma mensagem pode vir da Internet, passar pelos dispositivos do sistema de rede sem fio aquática, sendo comprimida e chegando ao dispositivo aquático final. Este dispositivo aquático gera uma mensagem de resposta que foi descomprimida no *gateway* e enviada de volta ao remetente remoto na Internet, por meio do protocolo ICMP tradicional. Os resultados mostram que dispositivos de redes de sensores aquáticas, que antes não conversavam com a Internet, podem ser monitorados remotamente. Mais ainda, o ganho na compressão é significativo, chegando a 93,75% no consumo energético, permitindo assim a criação da Internet das Coisas Aquáticas.

O restante deste trabalho é composto como a seguir. Na Seção 2 apresentamos os trabalhos relacionados. Na Seção 3 apresentamos a solução proposta, contendo o cenário considerado, a compressão de cabeçalhos e a descrição da implementação. Na Seção 4 apresentamos os testes e resultados. Finalmente, na Seção 5 apresentamos a conclusão e trabalhos futuros.

2. Trabalhos Relacionados

Existem diversas propostas realizadas com o objetivo de superar obstáculos encontrados para a comunicação sem fio no ambiente aquático. Protocolos de enlace, como Aloha [Vieira et al. 2006], modelos de mobilidade [Caruso et al. 2008], serviços e técnicas de localização [Vieira et al. 2010b, Erol et al. 2007b, Erol et al. 2007a, Erol et al. 2008] protocolos de roteamento como o Pressure Routing [Lee et al. 2010], GEDAR [Coutinho et al. 2014], oportunístico [Vieira 2012, Coutinho et al. 2016d], geográfico [Coutinho et al. 2016a], com controle de profundidade [Coutinho et al. 2013], baseados em centralidade [Coutinho et al. 2016c] com controle para nós dormirem [Coutinho et al. 2015], além de técnicas para economizar energia [Coutinho et al. 2016b], existem para as redes de sensores aquáticas. Apesar dos evidentes avanços que estes trabalhos trazem, nenhum trata a questão de criar a Internet das Coisas Aquáticas e/ou permitir que mensagens da Internet trafeguem pelas redes de sensores aquáticas.

Existem propostas para o acesso remoto a redes sem fio aquáticas (RSSFAs) e a seus dispositivos (sensores e nós). Porém, ainda não há um cenário que permita que RSSFAs possam ser remotamente reconfiguradas [Sun and Melodia 2013]. De fato, [Sun and Melodia 2013] propõem uma nova arquitetura que visa interoperabilidade com a arquitetura TCP/IP, permitindo o acesso remoto a uma rede aquática através da Internet. A proposta tem como características de destaque a inclusão de uma camada de adaptação à pilha de protocolos TCP/IP tradicional e a presença de um dispositivo na superfície, responsável pela comunicação com a Internet. Além disso, o trabalho inclui a instalação e execução de *softwares* de *driver* a nível de *Kernel* e a nível de usuário para viabilizar o seu funcionamento. Trata-se de uma solução promissora que constitui um passo importante para o alcance da Internet das Coisas Aquáticas que pode colaborar com o nosso trabalho. No entanto, no presente trabalho, nós não propomos uma nova arquitetura, mas sim uma versão com compressão para o protocolo ICMP. Essa alteração não exige a instalação de *drivers* adicionais ou de aplicações executadas a nível de *Kernel*. Além disso, com as alterações propostas, RSSFAs contariam com suporte da pilha de protocolos da arquitetura TCP/IP e, assim, a reconfiguração poderia ser feita por aplicações como SSH e FTP. Mais ainda, além da reconfiguração, será possível também realizar o monitoramento utilizando ferramentas como *ping* e *traceroute*, úteis para planejamento e avaliação de desempenho de rotas.

O *6LoWPAN* [Kushalnagar et al. 2007], citado na introdução, propõe um princípio de compressão que servirá de inspiração para este trabalho. A ideia principal do *6LoWPAN* é fragmentar os longos pacotes do protocolo IPv6, em quadros menores, realizando também a compressão dos cabeçalhos das mensagens. A compressão toma como base os valores mais comuns que aparecem nos campos de uma mensagem, permitindo que seja reduzido o comprimento (em *bits*) de cada um dos campos.

Outros protocolos bem conhecidos também foram adaptados para redes limitadas, como RSSFAs. Por exemplo, [Choi et al. 2009] faz a adaptação do protocolo *Simple Network Management Protocol* - SNMP para redes pessoais e de baixo consumo. Trata-se de uma extensão modificada que torna possível a transmissão de mensagens do protocolo SNMP, operando, inclusive, sobre o 6LoWPAN, e que alcançou resultados positivos consideráveis. [Ng et al. 2013] também apresentam uma adaptação de protocolo, desta vez,

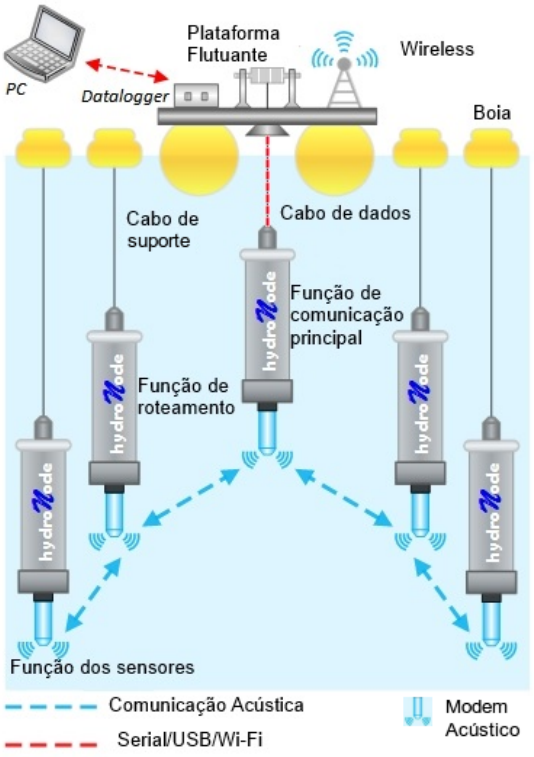


Figura 1. Cenário referência para este trabalho.

voltada ao ambiente de redes aquáticas. Os autores consideram os diversos obstáculos e características do cenário e trazem uma versão do protocolo MAC com *handshaking* assíncrono, conhecida como *Bidirectional-Concurrent MAC with Packet Bursting* (Bic-MAC). Note que, o foco do nosso trabalho é construir um bloco base para que diversos protocolos possam ser utilizados de forma indiscriminada entre Internet e RSSFA. Um dos exemplos adotados nesse trabalho é o monitoramento de dispositivos através da Internet, por meio do uso de mensagens mais curtas. A consequência natural dessa abordagem é uma melhor taxa de entrega de quadros na camada de enlace.

3. Solução Proposta

Nesta seção apresentaremos (i) o cenário considerado; (ii) nossa proposta de compressão de cabeçalhos e (iii) o Pingo d'água, uma adaptação do ICMP para monitoração de elementos em RSSFA.

3.1. Cenário Considerado

A Figura 1 apresenta um cenário típico de RSSFA. Em aplicações deste tipo, diversos nós sensores captam informações do meio. Comumente os nós se comunicam entre si. Estes nós sensores utilizam modens acústicos limitados para esta comunicação. Parte dos dados é recolhida por uma plataforma flutuante, que envia estes dados para posterior análise.

Neste trabalho, o modem acústico apresenta uma limitação de *buffer* de dados de apenas 24 bytes [Almeida et al. 2016]. Essa limitação é observada adiante, na proposta de compressão. Em outras palavras, desejamos alcançar uma compressão na qual as mensagens tenham, no máximo, 24 bytes de comprimento. Com relação a outros cabeçalhos que

podem compor a mensagem (cabeçalhos das outras camadas da arquitetura TCP/IP, por exemplo), não serão propostas alterações. Acreditamos que as soluções de compressão existentes na literatura, como o próprio *6LoWPAN*, poderão ser utilizadas em conjunto.

A Figura 1 também apresenta a *Datalogger Board*, um dispositivo que age como *gateway* de comunicação com o mundo exterior. Esse *gateway* é o dispositivo responsável pela comunicação dos dispositivos do ambiente aquático com os demais dispositivos terrestres, que operam segundo a pilha de protocolos da arquitetura TCP/IP. A comunicação com os dispositivos terrestres pode ocorrer, por exemplo, via cabo ou através de uma rede sem fio. O *gateway* está ligado via cabo a um nó sensor que, por sua vez, se comunica com os demais nós sensores pelo canal acústico. Por isso, a compressão das mensagens vindas da Internet para os dispositivos aquáticos será realizada no *gateway*, e nele também ocorrerá a descompressão das mensagens que fluirão no sentido oposto.

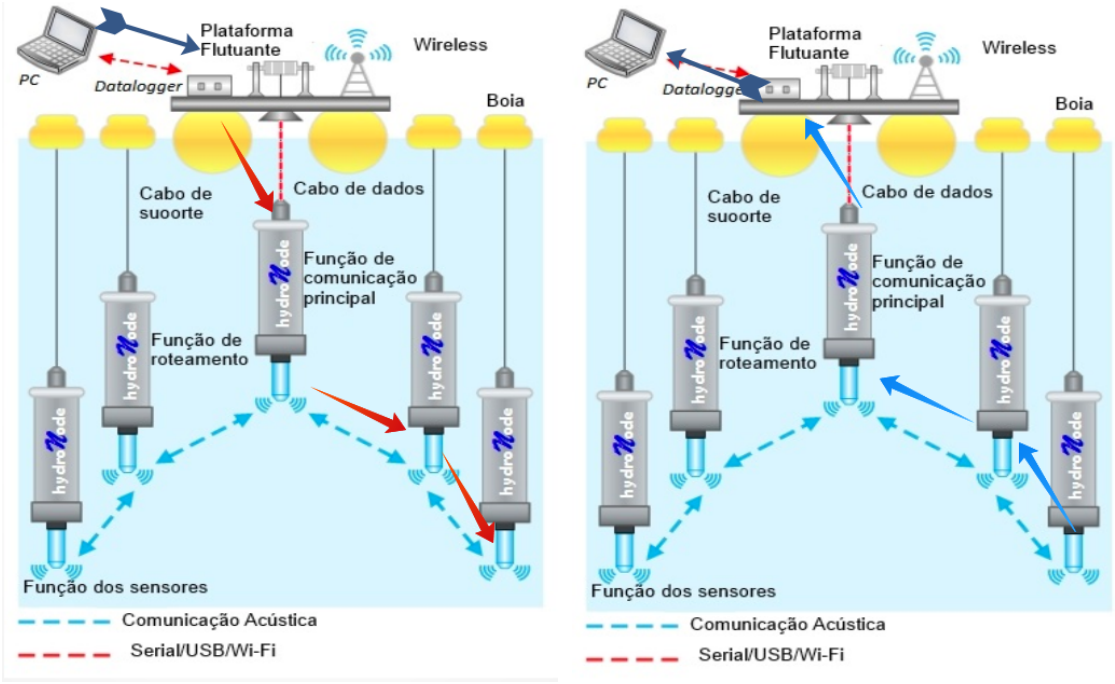
Seguindo esta especificação, temos como objetivo (mas não limitados a este) executar o comando Ping, composto pelas mensagens do tipo *Echo Request* e *Echo Reply* do protocolo ICMP, em qualquer computador na Internet. O fluxo de execução do comando pode ser ilustrado em duas etapas: a primeira, onde a requisição (*Echo Request*) flui de um dispositivo terrestre para os dispositivos aquáticos, mostrada na Figura 2(a); e a segunda, onde o nó destino da requisição gera a resposta (*Echo Reply*) e esta é enviada, fluindo no sentido contrário, como na Figura 2(b), com as respectivas compressão e descompressão sendo realizadas pelo *Gateway*.

3.2. Compressão de Cabeçalhos

Três campos estão presentes em toda mensagem do tipo ICMP, são eles: Tipo (*Type*), Código (*Code*) e *Checksum*. Os dois primeiros possuem 8 bits cada e o terceiro, 16. A composição e disposição do restante de uma mensagem ICMP depende dos valores do Tipo e do Código. Ou seja, dependem de qual mensagem do protocolo está sendo enviada. As mensagens trocadas pelo comando Ping possuem também os campos Identificador (*Identifier*) e Número de Sequência (*Sequence Number*), cada um tendo 16 bits, e o restante é composto pelos dados que serão "ecoados" pelos dispositivos, sendo também chamados de Carga (*Payload*) e tendo um tamanho usual de 56 bytes. Assim, é totalizado um tamanho usual de 64 bytes para as mensagens do comando Ping.

Uma ilustração do cabeçalho ICMP descrito para as mensagens *Echo Request* e *Echo Reply* [Postel 1981, Conta and Gupta 2006] pode ser vista na Figura 3 [Postel 1981]. Em cima deste cabeçalho, aplicamos um princípio de compressão semelhante ao existente no 6LowPAN. Assumiremos que os campos que compõem as mensagens possuirão com mais frequência alguns valores do que outros, e esses valores mais frequentes serão considerados valores comuns. Além disso, também iremos reduzir o tamanho dos campos que não possuem valores comuns (*Checksum*, Identificador, Número de Sequência e Dados), em função do novo cenário em que o protocolo será aplicado.

No corrente trabalho, consideramos como valores comuns os valores mais presentes nas mensagens *Echo Request*, *Echo Reply* e *Destination Unreachable*. Em nossos testes iniciais, as mensagens trocadas pelos dispositivos terão apenas os cabeçalhos e dados de mensagens ICMP. Os demais cabeçalhos, que fazem parte de uma mensagem na pilha de protocolos TCP/IP, não serão aqui incluídos. Dessa maneira, a compressão dos campos de uma mensagem ICMP será feita como a seguir:



(a) Comando Ping: *Echo Request* (b) Comando Ping: *Echo Reply*

Figura 2. Mensagens ping e pong através da Rede.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Identifier          |       Sequence Number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Data ...
+-+-+-+-+-
```

Figura 3. Cabeçalho das mensagens *Echo Request* e *Echo Reply* [Postel 1981].

- **Tipo** (*Type*): Para as mensagens do comando Ping, este campo recebe os valores 8 (*Echo Request*) e 0 (*Echo Reply*), no caso do protocolo ICMP. No caso do protocolo ICMPv6, os valores são, respectivamente 128 e 129, correspondentes às mesmas mensagens. Para a mensagem *Destination Unreachable*, o campo recebe os valores 3 e 1, para os protocolos ICMP e ICMPv6, respectivamente. Podemos assumir três valores comuns para o campo Tipo, um para cada mensagem mencionada, desprezando a diferenciação entre as versões do protocolo ICMP e definindo o valor deste campo como 2 bits. Assim, para 01 teremos *Echo Request*, para 00, *Echo Reply* e para 10, *Destination Unreachable* e o valor 11 será utilizado para indicar que este campo não está no seu valor comum.
- **Código** (*Code*): Este campo recebe sempre o valor 0 para as mensagens do comando Ping. No caso de uma mensagem do tipo *Destination Unreachable*, o valor 0 indica que não foi encontrada uma rota para o destino. Por isso, achamos prudente assumir 0 como sendo o valor comum do campo Código. Assim, quando em 0, o campo Código está no seu valor comum e, quando em 1, o campo está

com um valor diferente do comum.

- ***Checksum***: Ao invés do tamanho original de 16 bits, dado que a mensagem terá um tamanho menor do que o original, podemos reduzir o *Checksum* para 8 bits, deixando-o, assim, com metade do tamanho inicial.
- **Identificador** (*Identifier*): Uma vez que é baixa a quantidade de dispositivos que se comunicarão em baixo d'água e que, em uma mesma rede, não é viável que muitas mensagens estejam circulando no ambiente, podemos reduzir a 8 bits o identificador. Dessa forma, uma rede aquática funcionará com até 256 processos externos tentando acessá-la simultaneamente.
- **Número de Sequência** (*Sequence Number*): Pelas mesmas razões do item anterior e dado que uma característica do cenário aquático é a alta latência para a troca de mensagens, esta não pode ocorrer com muita frequência. Portanto, reduzimos este campo a 5 bits.
- **Dados** (*Payload*): Dada a necessidade de que as mensagens sejam curtas, a quantidade de dados a circular pela rede não precisa nem pode ser muito alta. Por isso, deixamos o campo de Dados com um tamanho padrão de 8 bits.

Temos assim uma redução de 40 bits – de 64 para 24 bits – no tamanho do cabeçalho e de aproximadamente 55 bytes – de usuais 56 para 1 byte – na parte de dados. A Figura 4 apresenta a nova estrutura da mensagem que circulará na rede subaquática.

Um campo com valor não comum terá o seu valor original concatenado ao final da mensagem. Assim, caso algum campo não tenha o valor comum, basta que o protocolo localize ao fim da mensagem o valor correto. Nosso protocolo preserva a ordem dos campos concatenados, tomando como referência sua posição original no cabeçalho.

A compressão do Identificador é feita através de uma espécie de mapeamento. O *gateway* é encarregado de manter uma relação entre um Identificador comprimido e o Identificador real correspondente. O *gateway* também cuida para que não haja repetição de Identificadores comprimidos.

Ao receber um identificador de 16 bits, o *gateway* escolhe o byte menos significativo deste identificador e, caso não exista um identificador comprimido com este valor, ele o armazena. Caso já exista, ele escolhe a parte mais significativa e repete o procedimento. Se ocorrer de também existir um identificador comprimido com o valor do byte mais significativo do campo original, o *gateway* passa a procurar sequencialmente por novos valores. Se não houver mais identificadores de 8 bits disponíveis, a transmissão é cancelada. Ao realizar o processo de descompressão, a relação entre o identificador real e o comprimido é descartada – tornando livre o valor que estava sendo utilizado – e uma nova relação é criada caso o mesmo identificador volte a aparecer. Durante os processos de compressão e descompressão, o *checksum* é recalculado, pois cada processo é responsável por gerar uma nova mensagem com tamanho e valores diferentes. A compressão do número de sequência é feita pegando-se a parte menos significativa do número de sequência original, tarefa que também cabe ao *gateway*.

3.3. Pingo d'água

Por fim, precisamos de uma maneira de tornar possível que, a partir de um terminal qualquer, seja possível executar o comando Ping tendo como destino um dispositivo subaquático. Esse dispositivo não está acessível pela Internet e, assim, não tem um

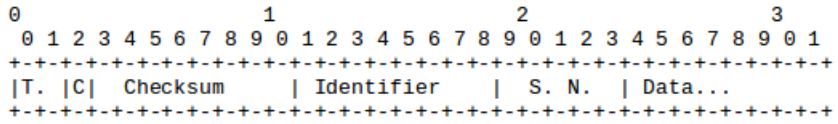


Figura 4. Cabeçalho ICMP Comprimido.

endereço IPV4 ou IPV6 tradicional. Além disso, pretendemos que não seja necessária uma alteração na pilha de protocolos ou em alguma aplicação já existente.

Como o cenário de rede aquática considerado neste trabalho não está em conformidade com a pilha de protocolos TCP/IP, é necessária uma maneira que torne possível identificar quando uma mensagem deve ou não ser encaminhada a uma rede aquática. Devemos também determinar o dispositivo destinatário no interior desta rede. Por exemplo, no cenário deste trabalho o *gateway* se conecta atualmente a um computador através da comunicação USB e, portanto, nesta configuração não tem o seu próprio endereço IP. Assim, cabe ao computador determinar se uma mensagem que chega até ele deve ou não ser encaminhada à rede aquática.

O envio de mensagens Ping é feito utilizando a ferramenta Ping normal, encontrada em diversos sistemas operacionais como Linux e Windows, e que pode ser executada via terminal. Dessa forma, em qualquer lugar da Internet se pode enviar uma mensagem para os dispositivos aquáticos. A ferramenta Ping tem uma funcionalidade que permite alterar os dados que serão "ecoados" na rede. Ou seja, é possível modificar o campo Carga (*Payload*). Para isto, basta executar o comando com o parâmetro "-p" seguido pelos valores com que se deseja preencher o campo, em hexadecimal, tendo um máximo de 16 bytes configuráveis. Utilizamos esta funcionalidade para possibilitar a identificação da mensagem que deveria ser enviada a rede aquática. Criamos um padrão que indica que a mensagem deve ser encaminhada à rede aquática, contendo também o ID do dispositivo destinatário. Portanto, para disparar um *ping* tendo como destino um dispositivo aquático, deve ser acrescentado o parâmetro "-p" com o seguinte padrão: "0FC000*ii*" onde cada *i* equivale a um dígito hexadecimal do ID do dispositivo, sendo o *i* mais à direita o menos significativo. Por exemplo, a execução do comando Ping tendo como destino um dispositivo aquático cujo ID é 15 deve ser assim: `ping <addr> -p 0FC0000F`. Onde *addr* é o endereço IP do computador conectado ao *gateway* da rede aquática.

Desenvolvemos um *software*, que chamamos de *ICMPforwarder*[1], que fica em execução no computador *gateway*. Esta aplicação, desenvolvida utilizando a Linguagem C de programação, é encarregada de verificar o padrão nos dados e encaminhar a mensagem para a rede aquática. O *gateway*, por sua vez, ao executar o processo de compressão, adicionará o ID recebido no padrão ao campo de dados da mensagem comprimida. Assim, um nó aquático saberá se é o destinatário da mensagem. Esta medida possibilita a operação em redes que não oferecem nenhum tipo de suporte ao protocolo IP.

4. Testes e Resultados

Para realização dos testes foi montada uma topologia. A topologia consiste em um computador, o *gateway* e um nó aquático. O processador presente no *gateway* é um

[1]https://github.com/franciscomilagres/PingoDagua

MSP430F247 e o dos dispositivos aquáticos, um MSP430F2274, ambos fabricados pela Texas Instruments®. O computador utilizado para execução da aplicação *ICMPforwarder* possui um processador Intel® Core *i7*®, 8GB de memória RAM e o seu sistema operacional é o Ubuntu® 14.04 LTS.

4.1. Testes sem ter como destino um dispositivo aquático

Primeiro testamos o Pingo d'água para demonstrar que o *ping* tradicional continua funcionando. As Figuras 5 e 6 ilustram a execução de um comando *ping* em uma máquina remota, sem ter como destino um dispositivo aquático. Como podemos ver, o *ICMPforwarder* identificou que a mensagem não deve ser encaminhada para a rede aquática via porta USB (figura 6) do *gateway*. A mensagem *Echo Request* recebida não é respondida e, a máquina com nome Eufrates, utilizada nos testes (figura 5), indica corretamente que nenhum pacote enviado foi respondido por dispositivo aquático.

```
eufrates:~> ping belohorizonte.lecom.dcc.ufmg.br -c 1
PING belohorizonte.lecom.dcc.ufmg.br (150.164.2.83) 56(84) bytes of data.

--- belohorizonte.lecom.dcc.ufmg.br ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Figura 5. Execução do Ping em um computador.

```
Listening...
===4: ICMP - Received ===

0:  45 0 0 54 83 29 40 0 3f 1 82 bf 96 a4 6 25
10:  96 a4 2 53 8 0 7 bd 36 a8 0 1 80 7a 2f 58
20:  0 0 0 0 3d f4 d 0 0 0 0 0 10 11 12 13
30:  14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23
40:  24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33
50:  34 35 36 37
IPv4: hdr-size=20 pkt-size=84 protocol=1 TTL=63 src=150.164.6.37 dst=150.164.2.83
ICMP: type[8/0] checksum[1981] id[13992] seq[1]
Not for RS232...
```

Figura 6. Ping recebido e não redirecionado.

4.2. Testes tendo como destino um dispositivo aquático

Nas Figuras 7 e 8 temos o disparo de um comando *ping*. Desta vez, tendo como destino um dispositivo aquático (figura 7). O *ICMPforwarder*, como mostrado na Figura 8, reconhece o padrão na mensagem e a encaminha para a porta USB do *gateway*, através do protocolo RS232. Ele aguarda a resposta do *gateway* e assim que recebe a mensagem de resposta do dispositivo aquático, o *ICMPforwarder* a envia para o remetente da mensagem.

O parâmetro "-c" recebido pelo comando *ping* indica quantas mensagens *Echo Request* deseja-se enviar. Vários testes foram feitos. Para facilitar o registro e visualização, realizamos estas imagens mostrando os testes enviando apenas uma mensagem.

O *gateway*, durante a execução desta tarefa, realizou a compressão da mensagem e a enviou a um dispositivo que compõe um nó aquático suportando o formato comprimido do Ping. Este, respondeu ao primeiro, que descomprimiu e encaminhou ao computador conectado.

```
eufrates:~> ping belohorizonte.lecom.dcc.ufmg.br -c 1 -p 0fc00001
PATTERN: 0x0fc00001
PING belohorizonte.lecom.dcc.ufmg.br (150.164.2.83) 56(84) bytes of data.
64 bytes from belohorizonte.lecom.dcc.ufmg.br (150.164.2.83): icmp_seq=1 ttl=63 time=300 ms

--- belohorizonte.lecom.dcc.ufmg.br ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 300.653/300.653/300.653/0.000 ms
```

Figura 7. Disparo do Ping com o parâmetro "-p".

```
===4: ICMP - Received ===

0:  45 0 0 54 f5 6 40 0 3f 1 10 e2 96 a4 6 25
10:  96 a4 2 53 8 0 f3 8 36 bc 0 1 f7 7a 2f 58
20:  0 0 0 0 3 dc 6 0 0 0 0 0 f c0 0 1
30:  f c0 0 1 f c0 0 1 f c0 0 1 f c0 0 1
40:  f c0 0 1 f c0 0 1 f c0 0 1 f c0 0 1
50:  f c0 0 1
IPv4: hdr-size=20 pkt-size=84 protocol=1 TTL=63 src=150.164.6.37 dst=150.164.2.83
ICMP: type[8/0] checksum[62216] id[14012] seq[1]

Forwarding to RS232.
-DONE...
Waiting for COM answer...
===16: COM - Received ===
Forwarding answer to host...
-DONE...
----------------------
```

Figura 8. Ping recebido e encaminhado ao *Gateway*.

4.3. Avaliação de Desempenho

Avaliamos o Pingo d'água com relação ao consumo energético, quantidade de bytes transmitidos e perda de pacotes na rede.

O gráfico presente na Figura 9 mostra a diferença de consumo energético entre transmissões contendo as mensagens comprimidas e transmissões sem compressão. Os valores de dados e cálculos utilizados para construção deste gráfico são semelhantes aos do trabalho de [Almeida et al. 2016], utilizando um modem acústico transmitindo

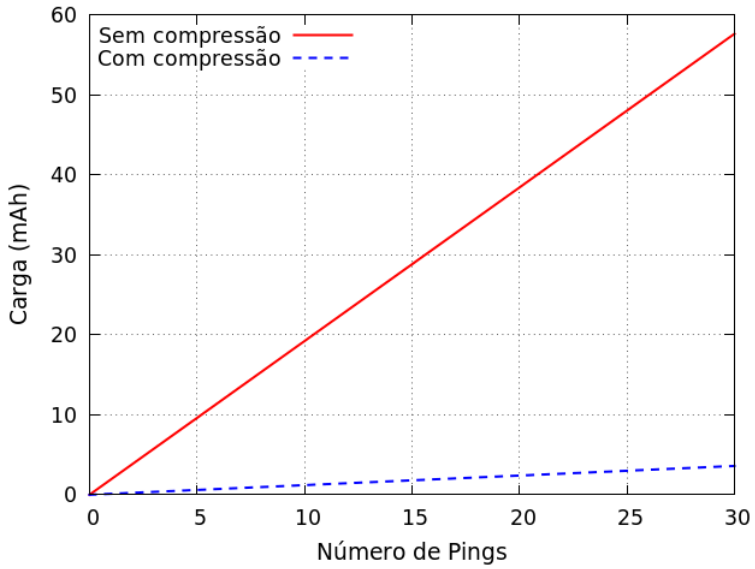


Figura 9. Consumo energético do tráfego na Rede Aquática.

na frequência de 33,8kHz. O consumo energético do modem é de 0,03 mAh por byte transmitido. A transmissão de uma mensagem ICMP com tamanho padrão e sem a nossa compressão exige uma quantidade de carga elétrica de 1,92 mAh, enquanto a transmissão de uma mensagem comprimida requer 0,12 mAh, consumindo 93,75% a menos. Ambas as curvas tem dependência linear com relação ao número de pings disparados. Mesmo assim, o consumo de energia com a utilização de nossa proposta é claramente menor.

Na Figura 10 está ilustrado o tráfego ICMP (em *bytes*) que ocorre na rede aquática, com a transmissão de mensagens ICMP e com a transmissão das mensagens comprimidas. A economia alcançada pela nossa compressão foi de aproximadamente 60 bytes por mensagem, dado que o tamanho padrão das mensagens ICMP trocadas pelo comando Ping é de 64 bytes. O desvio padrão (e intervalo de confiança) é desprezível pela quantidade de execuções e baixa variabilidade na compressão de dados tendo sido, por isso, omitido.

A Figura 11 apresenta a diferença na quantidade de pacotes perdidos entre transmissões compostas por mensagens comprimidas e transmissões compostas por mensagens ICMP no tamanho padrão, contando, neste caso, 3 pacotes por mensagem. O modem utilizado nos testes possui uma BER (*Bit Error Rate*) de 0,15% segundo a sua especificação, o que é responsável por gerar uma taxa de perda de pacotes de 4,69% para mensagens comprimidas e de 25,04% para as mensagens ICMP sem compressão, transmitidas em pacotes de, no máximo, 24 *bytes*.

5. Conclusão e Trabalhos futuros

Neste trabalho, apresentamos o Pingo d'água, uma versão com compressão de cabeçalho para o protocolo ICMP, desenvolvida para a ferramenta Ping funcionar em ambientes com comunicação aquática. Mensagens ICMP enviadas de qualquer computador da Internet podem alcançar os dispositivos aquáticos, e estes responderem de volta. A solução apresentada pode ser utilizada, por exemplo, em monitoração de lagos.

Através de um dispositivo intermediário, um *gateway*, as mensagens do proto-

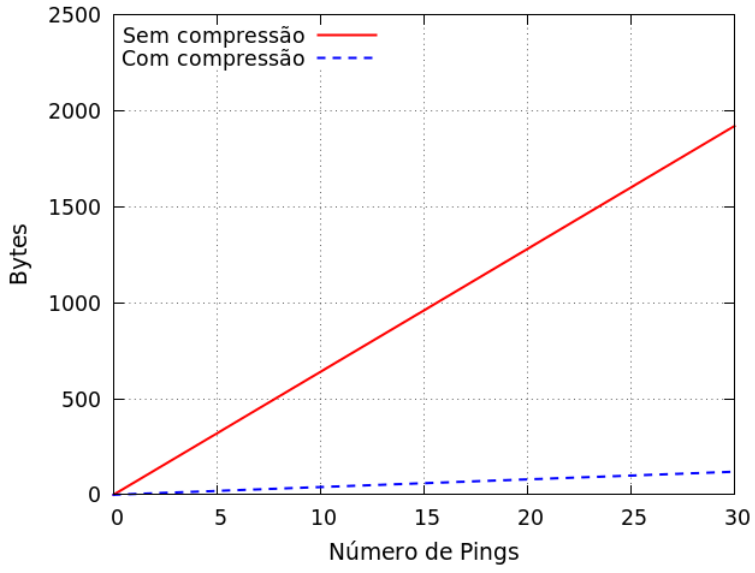


Figura 10. Tráfego ICMP na Rede Aquática.

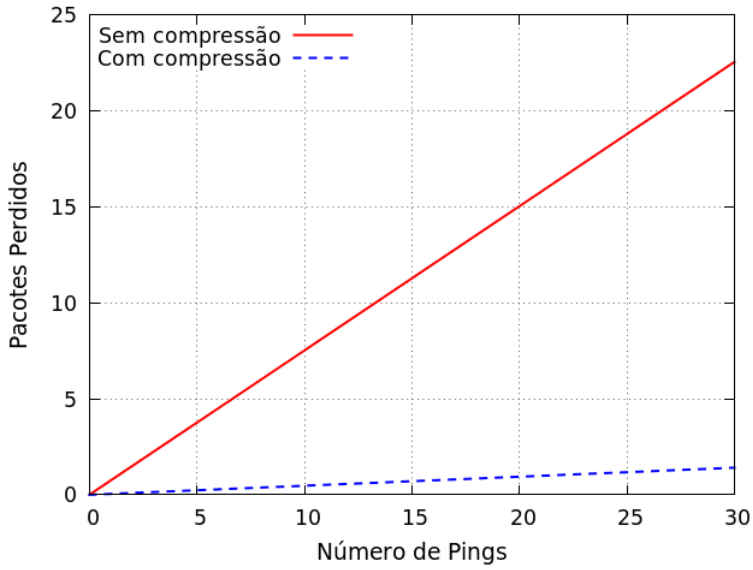


Figura 11. Perda de Pacotes na Rede Aquática.

colo ICMP enviadas de qualquer parte da Internet são comprimidas e transmitidas para os dispositivos aquáticos, cuja comunicação é limitada devido ao pouco poder computacional dos dispositivos e às características peculiares da comunicação aquática, como alta latência, baixa taxa de transmissão, alta taxa de perda de pacotes e outras, que se transformam em obstáculos para a criação da Internet das Coisas Aquáticas.

Em cenários como o descrito, é de fundamental importância possibilitar uma redução dos dados trafegados. Por isso, a compressão proposta pode ser um grande ganho se comparada à utilização do protocolo ICMP original, para o alcance da IoUT. Como proposto, há interoperabilidade tanto com o protocolo ICMP quanto com a versão ICMPv6.

Os resultados confirmam que os objetivos foram alcançados. Foi apresentado neste trabalho o envio de mensagens da Internet para dispositivos aquáticos a fim de facilitar a sua monitoração, sem necessitar modificar a arquitetura da pilha de protocolos TCP/IP ou mesmo ter que instalar drivers ou softwares adicionais. O trabalho apresentado tornou possível o acesso a uma rede sem fio aquática remotamente utilizando mensagens ICMP. Os testes mostraram a viabilidade e eficiência do protocolo proposto, permitindo a criação da Internet das Coisas Aquáticas.

Como trabalhos futuros podemos também operar com os demais cabeçalhos da pilha de protocolos da arquitetura TCP/IP, sobre o *6LoWPAN*. Também podemos utilizar uma versão do *6LowPAN* com baixo consumo de memória, como em [Santos et al. 2013]. Outras possibilidades são: dispensar a necessidade de parâmetros adicionais ao comando Ping em redes já atingíveis pela Internet; a integração com o protocolo DNS para o envio de mensagens por nome no lugar de identificadores; a adaptação do trabalho para operação com o *Wireshark*®; a inclusão de uma política de roteamento na rede, para testar a circulação das mensagens entre os nós; ou o uso de um mecanismo de endereçamento hierárquico com múltiplos saltos [Peres et al. 2016] que facilite o encaminhamento de mensagens.

Referências

Almeida, T. T., Bragança, L., Antônio, R. A. A. A., Lima, F., Vieira, M. A. M., Júnior, J. G. R., Vieira, L. F. M., and Nacif, J. A. (2016). Avaliação de desempenho na comunicação em redes acústicas aquáticas utilizando modems reais de baixo custo. In *Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP)*. SBC.

Caruso, A., Paparella, F., Vieira, L. F. M., Erol, M., and Gerla, M. (2008). The meandering current mobility model and its impact on underwater mobile sensor networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 221–225.

Choi, H., Kim, N., and Cha, H. (2009). 6LoWPAN-SNMP: Simple network management protocol for 6LoWPAN. In *High Performance Computing and Communications, 2009. HPCC '09. 11th IEEE International Conference on*, pages 305–313.

Conta, A. and Gupta, M. (2006). Internet control message protocol (ICMPv6) for the internet protocol version 6 (IPV6) specification.

Coutinho, R. W., Boukerche, A., Vieira, L. F., and Loureiro, A. A. (2015). Modeling and analysis of opportunistic routing in low duty-cycle underwater sensor networks. In *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 125–132. ACM.

Coutinho, R. W., Boukerche, A., Vieira, L. F., and Loureiro, A. A. (2016a). Geographic and opportunistic routing for underwater sensor networks. *IEEE Transactions on Computers*, 65(2):548–561.

Coutinho, R. W., Boukerche, A., Vieira, L. F., and Loureiro, A. A. (2016b). On the design of green protocols for underwater sensor networks. *IEEE Communications Magazine*, 54(10):67–73.

Coutinho, R. W., Boukerche, A. F., Vieira, L., and Loureiro, A. (2016c). A novel centrality metric for topology control in underwater sensor networks. In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 205–212. ACM.

Coutinho, R. W., Vieira, L. F. M., and Loureiro, A. A. F. (2013). DCR: Depth-controlled routing protocol for underwater sensor networks. In *2013 IEEE Symposium on Computers and Communications (ISCC)*, pages 453–458. IEEE.

Coutinho, R. W. L., Boukerche, A., Vieira, L. F. M., and Loureiro, A. A. F. (2014). GEDAR: geographic and opportunistic routing protocol with depth adjustment for mobile underwater sensor networks. In *2014 IEEE International Conference on Communications (ICC)*, pages 251–256. IEEE.

Coutinho, R. W. L., Boukerche, A., Vieira, L. F. M., and Loureiro, A. A. F. (2016d). Design guidelines for opportunistic routing in underwater networks. *IEEE Communications Magazine*, 54(2):40–48.

Erol, M., Vieira, L. F., Caruso, A., Paparella, F., Gerla, M., and Oktug, S. (2008). Multi stage underwater sensor localization using mobile beacons. In *Sensor Technologies and Applications, 2008. SENSORCOMM'08. Second International Conference on*, pages 710–714. IEEE.

Erol, M., Vieira, L. F., and Gerla, M. (2007a). Localization with Dive'N'Rise (DNR) beacons for underwater acoustic sensor networks. In *Proceedings of the second workshop on Underwater networks*, pages 97–100. ACM.

Erol, M., Vieira, L. F. M., and Gerla, M. (2007b). AUV-aided localization for underwater sensor networks. In *Wireless Algorithms, Systems and Applications, 2007. WASA 2007. International Conference on*, pages 44–54. IEEE.

Kushalnagar, N., Montenegro, G., and Schumacher, C. (2007). IPv6 over low-power wireless personal area networks (6lowpans): overview, assumptions, problem statement, and goals. Technical report.

Lee, U., Wang, P., Noh, Y., Vieira, L. F. M., Gerla, M., and Cui, J.-H. (2010). Pressure routing for underwater sensor networks. In *INFOCOM 2010. The 29th Conference on Computer Communications. IEEE*, pages 1676–1684.

Ng, H. H., Soh, W. S., and Motani, M. (2013). A bidirectional-concurrent mac protocol with packet bursting for underwater acoustic networks. *IEEE Journal of Oceanic Engineering*, 38(3):547–565.

Peres, B. S., Souza, O. A. d. O., Santos, B. P., Junior, E. R. A., Goussevskaia, O., Vieira, M. A. M., Vieira, L. F. M., and Loureiro, A. A. F. (2016). Matrix: Multihop address allocation and dynamic any-to-any routing for 6LoWPAN. In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 302–309. ACM.

Postel, J. (1981). [RFC792] Internet control message protocol.

Santos, E. R. d. S., Vieira, M. A. M., and Vieira, L. F. M. (2013). Routing IPv6 over wireless networks with low-memory devices. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2013 IEEE 24th International Symposium on*, pages 2398–2402. IEEE.

Sun, Y. and Melodia, T. (2013). The internet underwater: An ip-compatible protocol stack for commercial undersea modems. In *Proceedings of the Eighth ACM WUWNET*, WUWNet '13, pages 37:1–37:8, New York, NY, USA. ACM.

Viana, S. S., Vieira, L. F., Vieira, M. A., Nacif, J. A. M., and Vieira, A. B. (2015). Survey on the design of underwater sensor nodes. *Design Automation for Embedded Systems*, pages 1–20.

Vieira, L., Loureiro, A., Fernandes, A., and Campos, M. (2010a). Redes de sensores aquáticas. *XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, Gramado, RS, Brasil*, 1:199–240.

Vieira, L. F. M. (2012). Performance and trade-offs of opportunistic routing in underwater networks. In *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2911–2915. IEEE.

Vieira, L. F. M., Kong, J., Lee, U., and Gerla, M. (2006). Analysis of aloha protocols for underwater acoustic sensor networks. *Extended abstract from WUWNet*, 6.

Vieira, L. F. M., Lee, U., and Gerla, M. (2010b). Phero-trail: a bio-inspired location service for mobile underwater sensor networks. *IEEE Journal on Selected Areas in Communications*, 28(4):553–563.

Gerenciamento Hierárquico de Ambientes Inteligentes Utilizando SNMP

Denilson Rosa da Conceição, Jonas Ayres da Silva, Mathias Gheno Azzolini, Roben Castagna Lunardi, Rafael Pereira Esteves

Instituto Federal do Rio Grande do Sul (IFRS) – Campus Restinga
Porto Alegre – RS – Brazil

{drconceicao,jayres,mgheno}@restinga.ifrs.edu.br

{roben.lunardi, rafael.esteves}@restinga.ifrs.edu.br

Abstract. *Smart environments are designed to improve our daily lives by providing means to monitor a physical environment and react to the context in automatic and proactive ways. These environments are typically composed by a variety of sensors, actuators, and controllers interconnected by a network infrastructure. In order to allow interoperability with current Internet, such devices need to support a TCP/IP stack relying on IPv6 for addressing. Proper management of smart devices is required to allow a human operator to gather updated statistics and perform remote management over smart devices. In this paper, we investigate the use of SNMP, the de facto Internet network management protocol, in the management of smart environments. Since smart devices are typically resource constrained, we propose and develop an SNMP-based hierarchical management architecture to manage an environment composed of Arduino and Raspberry Pi boards. MIB extensions are developed and instrumented to support the proposed management architecture. Results demonstrate the feasibility of the SNMP protocol in managing smart devices without adding significant overheads.*

Resumo. *Ambientes inteligentes são projetados para melhorar nossa vida diária oferecendo meios para monitorar um ambiente físico e reagir ao contexto de forma automática e pró-ativa. Estes ambientes são tipicamente compostos de uma variedade de sensores, atuadores e controladores interconectados por uma infraestrutura de rede. Para permitir a interoperabilidade com a Internet atual, estes dispositivos precisam suportar uma pilha TCP/IP e utilizar IPv6 para endereçamento. O gerenciamento adequado de dispositivos inteligentes é necessário para permitir que um operador humano obtenha estatísticas atualizadas e realize o gerenciamento remoto desses dispositivos. Neste artigo, se investiga o uso do SNMP, o protocolo de gerenciamento de redes padrão da Internet, no gerenciamento de ambientes inteligentes. Uma vez que dispositivos inteligentes normalmente possuem recursos limitados, uma arquitetura de gerenciamento hierárquica baseada em SNMP é proposta e desenvolvida com o objetivo de gerenciar um ambiente composto de placas Arduino e Raspberry Pi. Extensões de MIB são desenvolvidas e instrumentadas para viabilizar a arquitetura de gerenciamento proposta. Resultados demonstram a viabilidade do protocolo SNMP no gerenciamento de dispositivos inteligentes sem adicionar sobrecarga significativa.*

1. Introdução

A Internet das Coisas (IoT) viabilizou uma série de aplicações para facilitar diversos aspectos do cotidiano [Atzori et al. 2010]. Exemplos de aplicações IoT incluem *e-health*, domótica, logística, transporte, entre outras. Tecnologias IoT podem ser aplicadas na implementação dos chamados *ambientes inteligentes*. Ambientes inteligentes são capazes de obter informações sobre o ambiente e de seus usuários de forma a proporcionar uma melhor experiência [Cook and Das 2007]. Uma casa inteligente, por exemplo, pode contribuir para a redução no consumo de energia desligando automaticamente luzes e equipamentos ociosos enquanto que uma fábrica inteligente pode monitorar em tempo real a produtividade de forma a identificar gargalos na produção.

Ambientes inteligentes são materializados através de dispositivos como sensores, atuadores e controladores interconectados por uma infraestrutura de comunicação. Sistemas de identificação por radiofrequência *(RFID - Radio Frequency Identification)* e redes de sensores sem fio *(WSN - Wireless Sensor Networks)* são exemplos de tecnologias IoT que são tipicamente utilizadas na implementação de ambientes inteligentes [Atzori et al. 2010] [Gubbi et al. 2013]. A comunicação dos dispositivos em um ambiente inteligente pode ser feita através de tecnologias como ZigBee, Bluetooth, WiFi, LoraWan, 3G/4G/LTE e, em casos específicos, Ethernet [Santos et al. 2016] [Gubbi et al. 2013].

Um aspecto importante nesse contexto diz respeito ao gerenciamento de ambientes inteligentes e dos dispositivos que fazem parte dos mesmos. O gerenciamento de ambientes inteligentes envolve principalmente a monitoração dos dispositivos, incluindo o estado de operação bem como os valores obtidos pelos sensores, e a configuração remota de dispositivos para permitir, por exemplo, que um operador humano ligue um equipamento (ex: ar-condicionado) antes de chegar na sua residência. Atualmente esse gerenciamento é feito através de soluções proprietárias e/ou protocolos específicos.

Neste artigo, se investiga o uso do protocolo de gerenciamento padrão da Internet, o SNMP *(Simple Network Management Protocol)* [Case et al. 1990] no gerenciamento dos dispositivos de um ambiente inteligente. A motivação para essa abordagem é permitir que plataformas de gerenciamento baseadas em SNMP possam ser rapidamente adaptadas para gerenciar um ambiente inteligente. Além disso, diferentes ambientes (ex: uma casa, uma indústria, uma fazenda) poderiam ser operados através de uma mesma interface de gerenciamento.

O principal desafio em adequar o SNMP como protocolo de gerenciamento para ambientes inteligentes é lidar com as restrições de capacidade dos dispositivos tipicamente utilizados nesse tipo de ambiente. Embora existam propostas que realizam uma série de adaptações e otimizações para permitir que um agente SNMP rode em um dispositivo IoT [Sehgal et al. 2012] elas normalmente dependem de *software* ou bibliotecas específicas que nem sempre estão disponíveis ou podem ser adaptadas para qualquer dispositivo IoT. Neste artigo, adota-se uma abordagem alternativa através de um *gateway* IoT consistindo de um dispositivo de maior capacidade que recebe e envia mensagens SNMP de/para uma aplicação de gerenciamento e interage diretamente com os dispositivos de menor capacidade. Dessa forma, os dispositivos finais (sensores e atuadores) não ficam sobrecarregados e a arquitetura padrão SNMP [Harrington et al. 2002] é mantida. Adicionalmente, com o objetivo de demonstrar a viabilidade da proposta, foi desenvolvido um sistema de gerenciamento de objetos inteligentes baseado na arquitetura proposta.

Além desta seção introdutória, o artigo é composto de mais 5 seções. Na seção 2 são discutidos os principais trabalhos relacionados ao gerenciamento de ambientes IoT. A arquitetura de gerenciamento para ambientes inteligentes proposta neste trabalho é apresentada e descrita na seção 3. Na seção 4 são discutidos aspectos de implementação do sistema de monitoramento de ambientes IoT. A arquitetura de gerenciamento proposta é avaliada na seção 5, através de um estudo de caso. Finalmente, na seção 6 são apresentadas as conclusões e os próximos passos do trabalho.

2. Trabalhos relacionados

Nos últimos anos, diversas pesquisas foram realizadas no contexto de Internet das Coisas. Grande parte da pesquisa em IoT está relacionada à definição de novas arquiteturas, soluções para comunicação entre dispositivos e estratégias de segurança para prevenir ataques. A seguir, nesta seção, serão apresentados os trabalhos mais relevantes relacionadas à nossa proposta.

Para resolver o problema de mobilidade em redes, Savolainen *et al.* [Savolainen et al. 2013] propõem a utilização do protocolo IPv6 em dispositivos de redes IoT. Devido ao fato do IPv6 provocar um consumo muito alto de memória para a sua utilização, o artigo propõe a utilização do IPv6 apenas nos *gateways* da rede de sensores.

Com o objetivo de integrar diferentes tecnologias de comunicação sem fio, tais como, WiFi, ZigBee, Bluetooth, Qin *et al.* [Qin et al. 2014] propõem uma arquitetura SDN (Software Defined Network) para redes IoT. Para alcançar este objetivo, os autores desenvolvem uma extensão para a arquitetura existente chamada de MINA (*Multinetwork INformation Architecture*), com um controlador SDN para atuar como *middleware* na camada de rede.

No contexto de comunicação e gerenciamento de dispositivos em redes IoT, Benamar *et. al.* [Benamar et al. 2014] realizam uma comparação de diferentes protocolos para serem utilizados em redes IoT. Por exemplo, no contexto de endereçamento IP, o trabalho propõe alternativas com menor consumo de memória e processamento através da utilização do padrão 6LoWPAN [Montenegro et al. 2007]. No contexto de gerenciamento dos dispositivos IoT, o artigo apresenta uma comparação entre os protocolos SNMP, LNMP (*Lightweight Network Management Protocol*), e as soluções propostas pelo grupo da IETF através da iniciativa COMAN (*Constrained networks and devices MANagement*), como o CoAP (*Constrained Application Protocol*) [Shelby et al. 2014].

Mais especificamente relacionado com a análise de tráfego IoT, Chen *et al.* [Chen et al. 2013] propõem um sistema de monitoramento e análise de trafego de dispositivos IoT. Neste trabalho, o sistema apresenta dados como, monitoramento de sensores e o estado dos dispositivos. Apesar de o sistema ser acessível pela Web ou por *smartphone*, não são utilizados protocolos padrão de gerência de redes. Sendo a obtenção dos dados realizadas diretamente através dos pinos de entrada e saída do *hardware* específico. Desta forma, a solução fica limitada ao hardware utilizado.

No trabalho de Chang *et al.* [Chang et al. 2012] apresentam uma plataforma para simulação de sistemas de IoT no contexto da Internet do Futuro (do inglês, *Future Internet*). Nesta simulação, o autor faz uma análise do roteamento das mensagens, que devido à arquitetura da rede, acaba por gerar um grande tráfego de informações para o coordenador de tráfego. Ainda utilizando esta simulação, é feita uma medição do tempo de atraso

na entrega das mensagens entre os componentes. Nesta medição pode ser percebido que à medida que o tempo e tráfego da simulação avançam, este tempo cresce rapidamente.

Um dos usos mais comuns de redes IoT é na implementação de redes de sensores inteligentes. Neste contexto, Ma *et al.* [Ma et al. 2016] propõem uma arquitetura de IoT separadas em quatro camadas: (i) camada de sensoriamento e controle; (ii) camada de troca de dados; (iii) camada de integração de informação; e (iv) camada de aplicação. Neste modelo, as informações só são trocadas entre as camadas subsequentes, sem haver acesso direto entre as camadas que não estão diretamente ligadas. A camada de sensoriamento e controle é a camada responsável por interagir com o mundo físico, transformando em sinais digitais informações analógicas e/ou interagindo com objetos físicos. A camada de troca de dados, por outro lado, é responsável apenas pela comunicação, utilizando diferentes tipos de meios e protocolos. A camada de integração da informação é responsável pelo processamento da informação e garantia de aspectos básicos de segurança. Por fim, a camada de aplicação combina dados coletados de diversos sensores que serão apresentados para o usuário final.

Para permitir o uso dos protocolos de gerenciamento SNMP e NETCONF em dispositivos de baixa capacidade, Sehgal *et al.* [Sehgal et al. 2012] desenvolveram versões simplificadas desse protocolos em dispositivos Atmel AVR Raven executando sobre o sistema operacional Contiki. Embora os autores demonstrem que as otimizações feitas permitem que agentes SNMP e NETCONF sejam executados em dispositivos IoT sem prejuízo significativo de desempenho, não é possível generalizar esse comportamento para outros dispositivos IoT com menor capacidade (ex: Arduino) e/ou que não tenham suporte adequado ao Contiki.

Analisando os trabalhos abordados até então, é possível notar que a maioria deles não exploram adequadamente o uso de protocolos padronizados no gerenciamento de ambientes IoT, ficando restritos a protocolos proprietários e/ou hardwares/softwares específicos, reduzindo a interoperabilidade que é desejável em ambientes inteligentes. Na próxima seção, é apresentada uma proposta de arquitetura baseada no protocolo padrão de gerenciamento da Internet, o SNMP, para contornar as limitações das soluções atuais de gerenciamento de ambientes IoT.

3. Arquitetura de gerenciamento de ambientes inteligentes baseada em SNMP

Nesta seção é apresentada a arquitetura de gerenciamento baseada no protocolo SNMP para dispositivos IoT que compõem um ambiente inteligente. Inicialmente, extensões feitas à base de informações de gerenciamento para viabilizar a arquitetura propostas são apresentadas. Em seguida, os componentes da arquitetura de gerenciamento proposta são detalhados.

3.1. Modelo de Dados

No protocolo SNMP, as informações referentes aos dispositivos gerenciados são estruturadas em uma ou mais bases de informações de gerenciamento (*MIB - Management Information Base*). No início da década passada o IETF definiu uma MIB para o gerenciamento de sensores físicos, a ENTITY-SENSOR-MIB [Bierman et al. 2002] que permite

monitorar os valores de diferentes tipos de sensores como corrente elétrica, potência, temperatura, umidade, entre outros.

Embora a ENTITY-SENSOR-MIB permita a monitoração de sensores de forma relativamente simples, ela apresenta algumas limitações para o gerenciamento de ambientes inteligentes. Por exemplo, existe a necessidade de uma melhor identificação dos sensores, uma vez que diferentes sensores do mesmo tipo podem estar presentes em um ambiente inteligente. A localização física de um determinado sensor também é importante pois permite que o gerente execute ações de gerenciamento localizadas, por exemplo, para desligar todos os equipamentos de uma sala. Ainda, distinguir se um determinado dispositivo é um sensor ou atuador permite determinar que ações podem ser executadas sobre o mesmo.

Para contornar as limitações atuais de gerenciamento foram feitas extensões à ENTITY-SENSOR-MIB com a inclusão de novos objetos para possibilitar o gerenciamento pleno de ambientes inteligentes. A Figura 1 mostra uma representação gráfica da ENTITY-SENSOR-MIB com os novos objetos incluídos. Em seguida, é apresentada uma breve descrição de cada novo objeto.

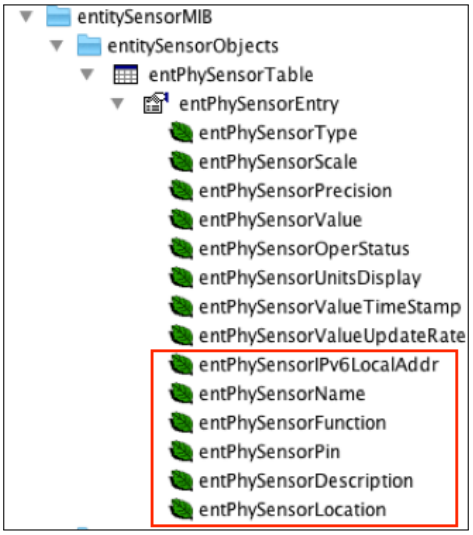


Figura 1. Extensões feitas à ENTITY-SENSOR-MIB

- *entPhySensorIPv6LocalAddr*: armazena o endereço IPv6 (link-local) do dispositivo;
- *entPhySensorName*: nome do dispositivo;
- *entPhySensorFunction*: função do dispositivo, sensor ou atuador:
- *entPhySensorPin*: pino onde o sensor está localizado. Útil para gerenciar dispositivos do tipo Arduino, que pode possuir vários sensores em pinos diferentes;
- *entPhySensorDescription*: breve descrição sobre a função do sensor (ex: luminosidade, temperatura);
- *entPhySensorLocation*: localização física do dispositivo.

3.2. Arquitetura de Gerenciamento Proposta

A arquitetura proposta neste trabalho baseia-se no conceito de arquitetura em camadas, conforme taxonomia definida por Ma *et al.* [Ma et al. 2016]. Para atender as necessidades do nosso trabalho, definiu-se *Camada de Sensoriamento e Controle* como a camada onde os dispositivos obtêm dados de sensores e configuram atuadores, apresentada na Figura 2 como *Dispositivos*. Entende-se por dispositivo o objeto microcontrolado, usualmente com restrições de processamento, capaz de utilizar um ou mais sensores (ex: luminosidade, temperatura, presença de algum tipo de gás, dentre outros) e/ou controladores (acionadores elétricos, emissor de sinais eletromagnéticos, motores de posicionamento, dentre outros). A *Camada de Troca de Dados* é responsável pela comunicação entre os dispositivos e os *Gateways*. Nesta arquitetura, os *Gateways* representam dispositivos com maior capacidade de *hardware*, capazes de realizar tarefas que demandam maior poder de processamento. A *Camada de Integração de Informação* é responsável pela consolidação das informações disponíveis nos *Gateways*. Por fim, a *Camada de Aplicação* é definida pela *Interface Web*, utilizada pelo administrador do sistema para visualizar as informações coletadas do ambiente.

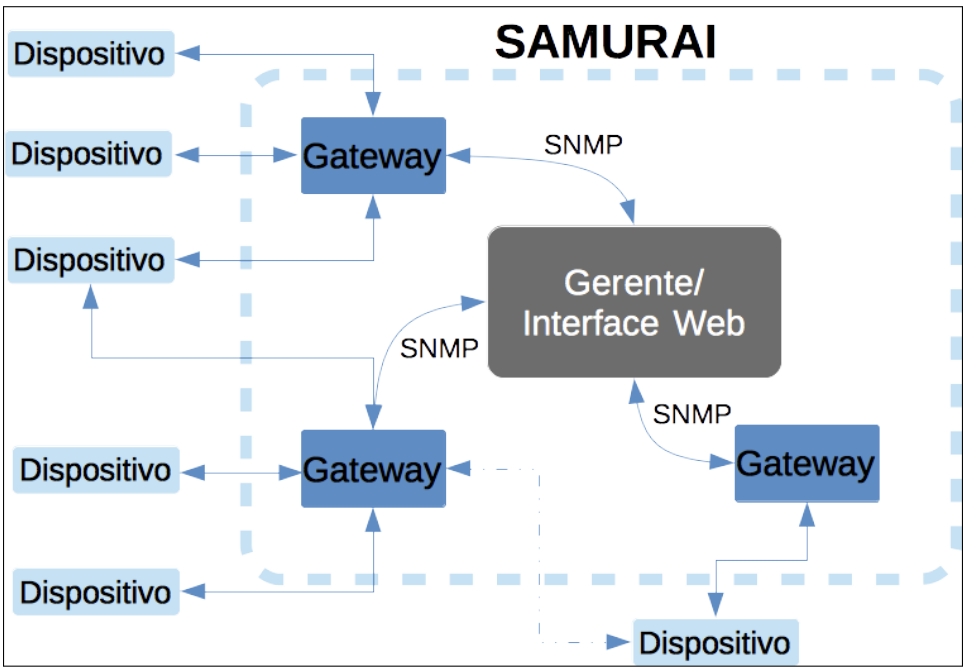


Figura 2. Arquitetura do SAMURAI

Devido às restrições de hardware (processamento, memória, energia, etc) dos dispositivos e pela possibilidade do uso de dispositivos heterogêneos, com diferentes restrições de *hardware*, optou-se pela utilização de *Gateways*. Desta forma, o processamento das informações, bem como a comunicação com os demais dispositivos de um mesmo domínio e com o Gerente ficam a cargo dos *Gateways*, diminuindo a carga de processamento nos dispositivos. Ainda, com a utilização de *Gateways*, permite-se a adoção de diferentes protocolos de comunicação, variando conforme as restrições de cada dispositivo. Portanto, a comunicação entre dispositivos e o *Gateway*, quando necessária, pode ser realizada utilizando protocolos com pouca exigência de processamento e memória (ex: Telnet e HTTP). Neste caso, utilizamos a premissa que os dispositivos estão configurados

para se comunicar em um VLAN dedicada e segura a transmissão dos dados. Sabe-se da importância dos requisitos de segurança, tais como autenticação, autorização, confidencialidade, e integridade durante a comunicação dos dispositivos. Todavia, estes fatores serão analisados em trabalhos futuros. Por outro lado, permite-se que dispositivos com maior capacidade de *hardware*, principalmente em canais de comunicação não confiáveis, realizem a transmissão utilizando protocolos que fazem uso de criptografia (ex: SSH e HTTPS). Adicionalmente, com o uso da arquitetura proposta, cada *Gateway* pode ficar responsável por monitorar um ou mais dispositivos. Desta forma, aumenta-se a resiliência do monitoramento dos dispositivos, diminuindo o efeito de falhas de um *Gateway*. Por fim, a arquitetura proposta permite a mobilidade dos dispositivos (por exemplo, sensores embarcados em equipamentos móveis), permitindo o ingresso e saída do dispositivo do domínio do *Gateway* (representado pela linha pontilhada na Figura 2).

O módulo Gerente monitora os *Gateways*, verificando a consistência da informação, visto que o mesmo dispositivo pode ser monitorado por mais de um *Gateway*. O Gerente, na arquitetura proposta, se comunica com os *Gateways* utilizando o protocolo SNMP. O SNMP foi adotado pela ampla popularidade, fácil implementação e possibilidades de adaptação às necessidades, conforme apresentado anteriormente na subseção 3.1 (Modelo de Dados). Pode-se destacar que o Gerente é impossibilitado de acessar diretamente os dispositivos, evitando, assim, sobrecarga destes dispositivos com maiores restrições de *hardware*.

Após a análise das informação obtidas pelo Gerente, o módulo Interface Web é capaz de processar e apresentar os dados dos dispositivos. Essas informações são processadas para a geração de gráficos com informações configuráveis por um operador humano, conforme a necessidade de monitoramento (ex: temperatura e luminosidade do ambiente, sobrecarga da CPU de um *Gateway*, dentre outros). Além disso, a interface permite o acionamento de dispositivos (ex: abertura de uma fechadura, ligar/desligar luzes, dentre outros). Este sistema de gerenciamento proposto no trabalho foi chamado de SAMURAI (Sistema de Monitoramento Unificado de Redes e Aplicações IoT).

Desta forma, o sistema SAMURAI se comunica diretamente com os *Gateways* para receber as informações e monitorar os dispositivos através do protocolo SNMP. Por sua vez, os *Gateways* se comunicam utilizando protocolo a ser definido para cada dispositivo, conforme as restrições de *hardware* do mesmo. Consequentemente, os agentes SNMP ficam nos *Gateways*, reduzindo o processamento, comunicação e, consequentemente, o consumo de energia dos dispositivos.

4. Implementação

Nesta seção, são abordados os aspectos de implementação do sistema, destacando as tecnologias utilizadas, o processo de desenvolvimento e os principais módulos do SAMURAI. Para a implementação do sistema foi utilizada a linguagem de programação Ruby [Ruby 2016], juntamente com os *frameworks* Ruby on Rails [Ruby on Rails 2016] e Daemons [Ruby Daemons 2016]. Adicionalmente, para a implementação da Interface Web, foi utilizada o *framework* bootstrap [Bootstrap 2016]. Ainda, para a geração de gráficos, foi utilizado a API Google Charts [Google 2016]. Por fim, para a comunicação com os dispositivos foi utilizado o protocolo SNMP (versão 2), bem como demais protocolos de rede como, IPv4 e IPv6.

O módulo Gerente/Interface Web, apresentado anteriormente na Figura 2, pode ser visto em detalhes (como submódulos separados) na Figura 3. O módulo da Interface Web (Figura 3) tem como objetivo prover uma interface de interação com o usuário através de um sistema Web. Desta forma, o usuário (Administrador do Sistema) pode cadastrar os dispositivos associados aos *Gateways* e cujos dados serão apresentados através de gráficos, listas e painéis. Vale destacar que todas as informações são coletadas e armazenadas em um banco de dados.

Por outro lado, o módulo Cadastrador (Figura 3) na aplicação tem como objetivo cadastrar os dispositivos que estão associados aos *Gateways*, os quais podem ser solicitados pela aplicação Web. Vale destacar que o mesmo dispositivo pode estar presente em mais de um *Gateway*, sendo o último o responsável por se comunicar com os dispositivos. Desta forma, a aplicação não se comunica diretamente com os dispositivos, pois todas as requisições são realizadas aos *Gateways* responsáveis por cada dispositivo. Esta forma de acesso aos dados reflete a arquitetura IoT proposta, conforme descrito na subseção 3.2 (Arquitetura de Gerenciamento Proposta).

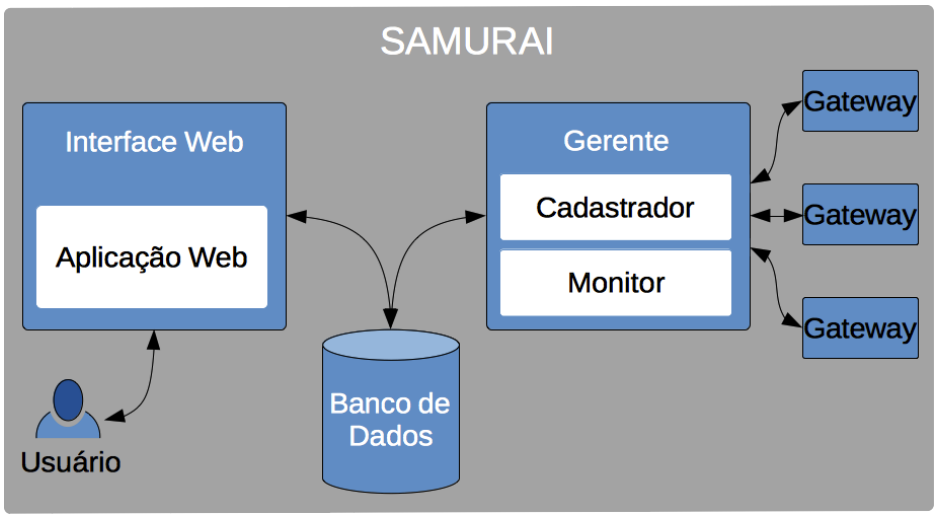


Figura 3. Implementação do Gerente/Interface Web

O módulo Monitor (Figura 3) tem como objetivo coletar dados dos equipamentos do sistema, tanto dos *Gateways* e dos dispositivos que estão a cargo dos mesmos, quanto dos equipamentos diretamente cadastrados pela interface Web do sistema. Ressalta-se que, atualmente, todas as informações são armazenadas em uma base de dados SQLite (em uma próxima versão pretende-se adotar um banco de dados NoSQL).

Na Figura 4 pode ser visualizado um dispositivo cadastrado, com IP do(s) *Gateway(s)* correspondente(s), bem como informações do dispositivo monitorado, tais como IP, tipo de sensor, valor correspondente aos dados monitorados, e sala onde o dispositivo foi cadastrado. Além disso, pode ser observado um gráfico da variação da luminosidade medida pelo dispositivo em função do tempo. Os gráficos a serem exibidos podem ser configurados pelo usuário para melhor demonstrar as informações. Por limitações de espaço, não serão apresentadas telas relativas ao cadastro de dispositivos, resumo do sistema (aba *Dashboard*) e demais funcionalidades.

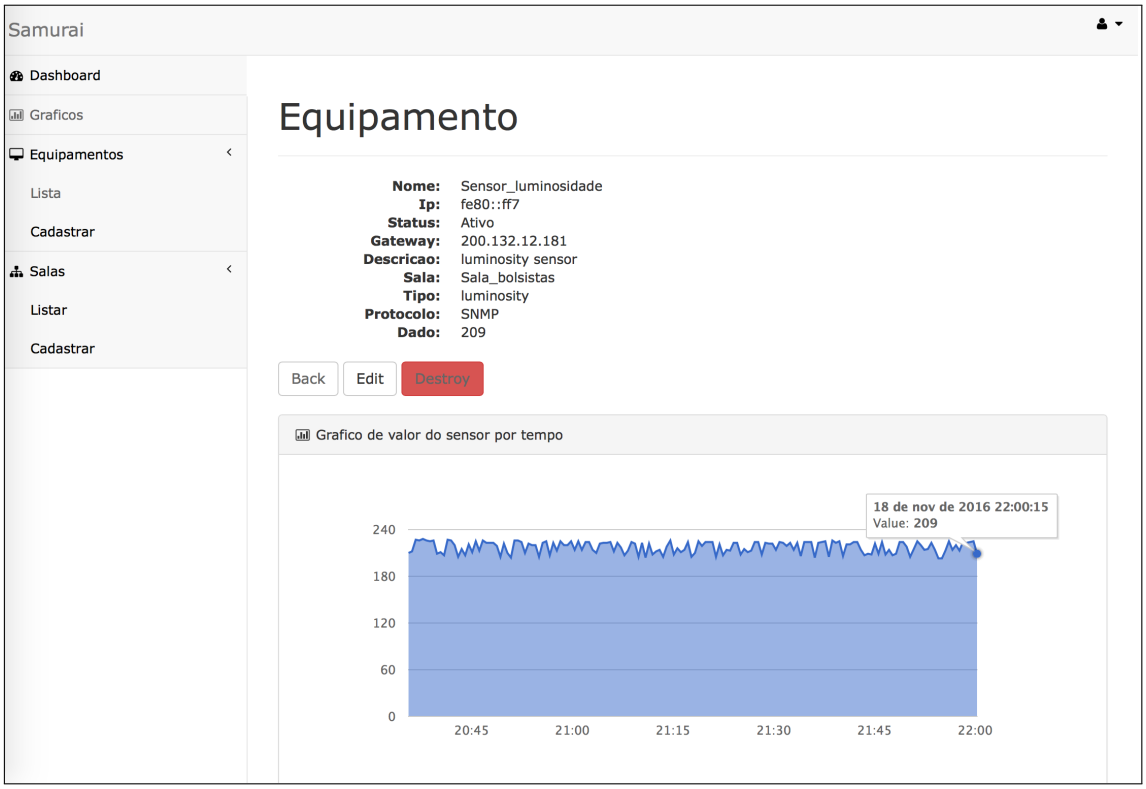


Figura 4. Interface Web do Sistema SAMURAI com dados de Sensor de Luminosidade

5. Estudo de caso

Para validar a arquitetura proposta, um estudo de caso foi conduzido com dispositivos reais em um cenário *indoor*. Os dispositivos são representados por placas controladoras Arduino Uno [Arduino 2016], cada uma equipada com um sensor de luminosidade e uma interface de rede Ethernet para comunicação. O *Gateway* é materializado através de um dispositivo Raspberry Pi 3 Modelo B [Raspberry Pi 2016] executando o sistema operacional Raspbian. O agente SNMP foi instalado no Raspberry Pi utilizando a suíte NET-SNMP [NET-SNMP 2016]. A ENTITY-SENSOR-MIB estendida foi instrumentada através de um módulo de MIB e incorporada ao agente SNMP. A comunicação entre o *Gateway* e os dispositivos é feita através de uma implementação Telnet simples. Dessa forma, as informações obtidas via Telnet são utilizadas para alimentar os objetos da ENTITY-SENSOR-MIB.

Foram utilizados três Arduinos Uno e um Raspberry Pi em nossos experimentos. Para coletar os dados, o Gerente dispara uma requisição *SNMP Walk*[1] para o agente SNMP instalado no *Gateway*. Essa requisição tem por objetivo recuperar as informações de todos os objetos armazenados na ENTITY-SENSOR-MIB para os três Arduinos gerenciados. Após receber receber a requisição SNMP, o *Gateway* recupera o valor dos sensores de luminosidade via Telnet e atualiza as informações da ENTITY-SENSOR-MIB. Por fim, o agente responde à requisição SNMP com as informações atualizadas dos dispositivos.

[1]Uma requisição SNMP Walk é constituída de uma série de requisições *get* e *get-next* que recuperam todos os objetos associados a um OID (*Object Identifier*) *específico.*

Do ponto de vista do Administrador do Sistema, os dados dos dispositivos e dos *Gateways* são facilmente visualizados através da interface do SAMURAI. Por exemplo, pode-se monitorar o valor de um sensor gerenciado pelo *Gateway*, utilizando o objeto da MIB *entPhySensorValue* (presente na ENTITY-SENSOR-MIB). No caso do "Gráfico de valor do sensor no tempo", como demonstrado na Figura 5, apresenta-se o valor do sensor durante o tempo (tempo de relógio do *Gateway*). No exemplo, pode-se observar que o dado monitorado (item "Dado" na Figura 5) - neste caso, valor correspondente à intensidade da iluminação obtida através de um sensor LDR (*Light Dependent Resistor*) - possui o valor 1012 (dez mil e doze). Pode-se observar, ainda, que o valor permanece praticamente constante durante o período que foi realizada a medição. Além disso, no "Gráfico de tempo de resposta do dispositivo" (Figura 5) pode ser observado o tempo de resposta das requisições do *Gateway* ao dispositivo, medida em milissegundos. Este tempo de resposta é calculado pelo início da conexão com o *Gateway* até a resposta do dispositivo. Neste gráfico, pode-se observar picos nos tempos de resposta, os quais usualmente representam os instantes nos quais o dispositivo está respondendo às requisições de dados solicitadas pelo *Gateway* via protocolos IPv6 e Telnet.

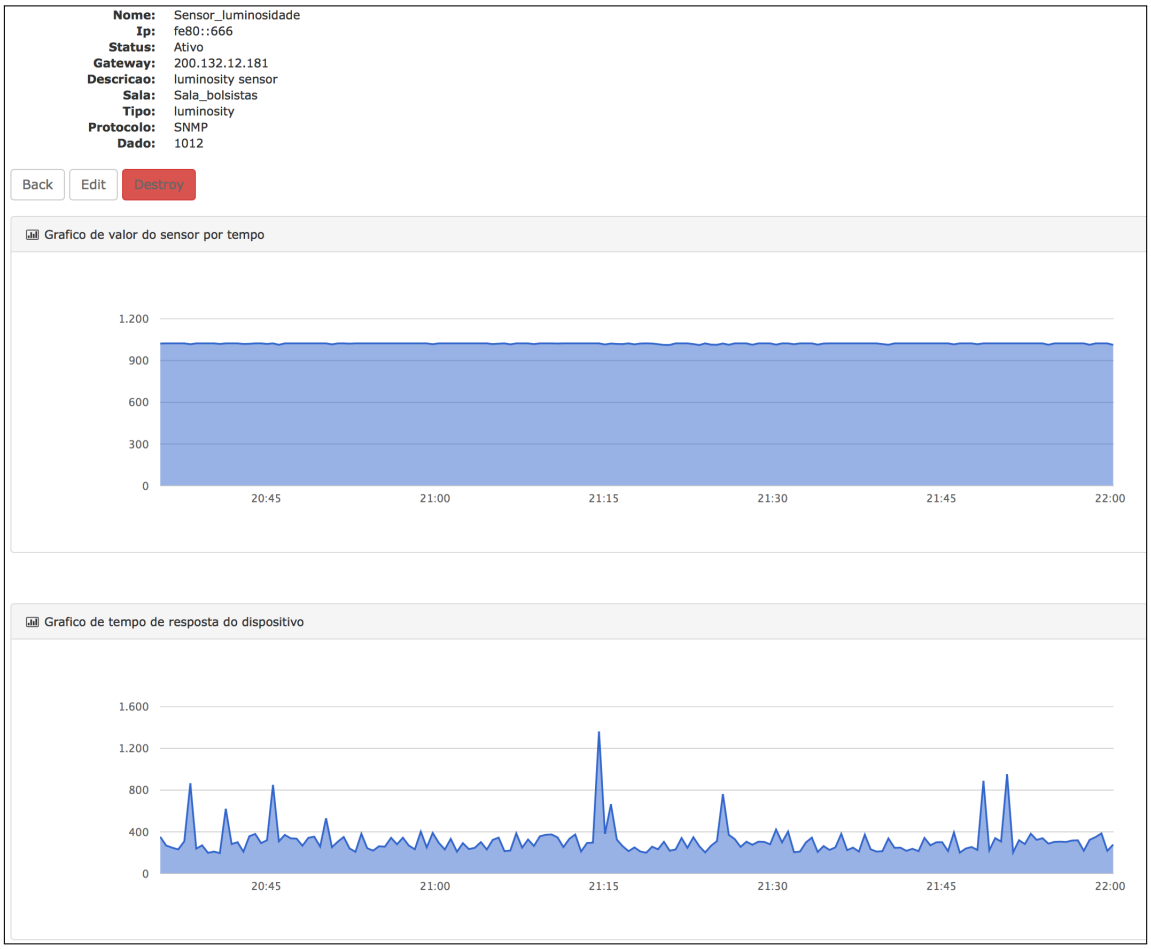


Figura 5. Monitoramento do Sensor pelo SAMURAI

Na Figura 6, pode ser observado o tempo de resposta do *Gateway* monitorado pelo SAMURAI. Este tempo de resposta, equivale ao tempo para que o *Gateway* receba o

retorno de todos os respectivos dispositivos (quando ativos). Consequentemente, o tempo de resposta do *Gateway* (também obtido em milissegundos) é maior que a resposta de um único dispositivo. Além disso, pode ser observado que existem picos de resposta em maior quantidade do que o dispositivo analisado. Isto ocorre pelo fato de que cada dispositivo pode responder às requisições do *Gateway* em tempo distintos. Desta forma, sempre que um dispositivo apresenta aumento no seu tempo de resposta, o tempo de resposta do *Gateway* também é alterado na exibição do sistema.

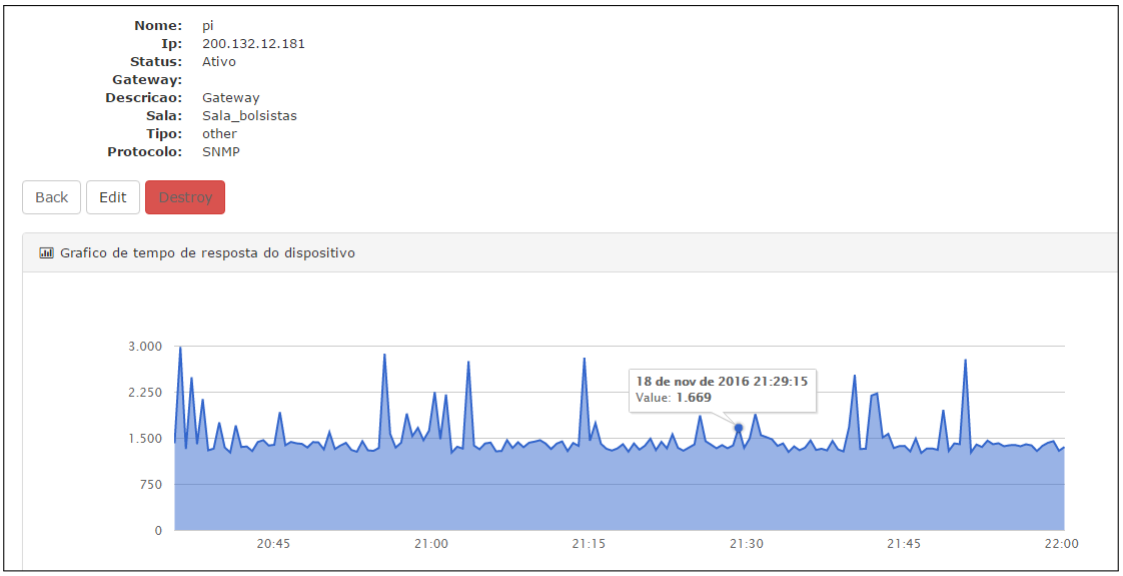


Figura 6. Monitoramento do *Gateway* pelo SAMURAI

Outro objetivo do estudo de caso é avaliar quantitativamente o desempenho da comunicação entre o *Gateway* e os dispositivos em relação ao tempo de resposta, tempo de CPU e utilização de memória, para verificar o impacto desta etapa no desempenho total da solução. O tempo de resposta nesse caso consiste do tempo que o *Gateway* leva para obter as informações dos dispositivos e atualizar os objetos da ENTITY-SENSOR-MIB. As medições referentes à tempo de CPU e utilização de memória são obtidas no *Gateway* através do utilitário *ps*. Os experimentos foram repetidos cerca de 80 vezes e os resultados são apresentados com intervalo de confiança de 95%.

A Figura 7 mostra o tempo de resposta para a comunicação entre o *Gateway* e os dispositivos finais para recuperar as informações de um, dois e três dispositivos, respectivamente. É possível observar que o tempo que o *Gateway* leva para recuperar as informações dos dispositivos finais (Arduino) cresce linearmente com o número de dispositivos gerenciados, o que é esperado uma vez que a quantidade de interações necessárias para recuperar os valores dos sensores e alimentar a ENTITY-SENSOR-MIB cresce com o número de dispositivos monitorados. No entanto, vale ressaltar que no pior caso (três dispositivos) esse tempo fica na ordem de 180 milissegundos por objeto, o que é consistente com o tempo de resposta acumulado no *Gateway* (Figura 6).

A Figura 8 apresenta a utilização de CPU para cerca de 260 requisições SNMP (*get* e *get-next*) de objetos da ENTITY-SENSOR-MIB. Essas requisições menores são derivadas das requisições *SNMP Walk* disparadas pelo módulo Gerente do SAMURAI.

É possível observar que a utilização de CPU se mantém abaixo de 4% para mais de 250 requisições SNMP, o que pode ser considerado baixo.

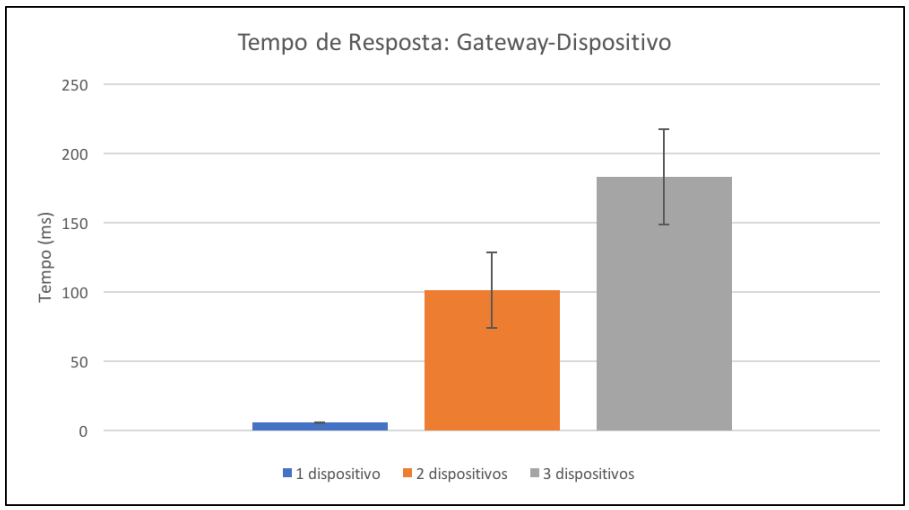


Figura 7. Tempo de resposta na entre o *Gateway* e os dispositivos

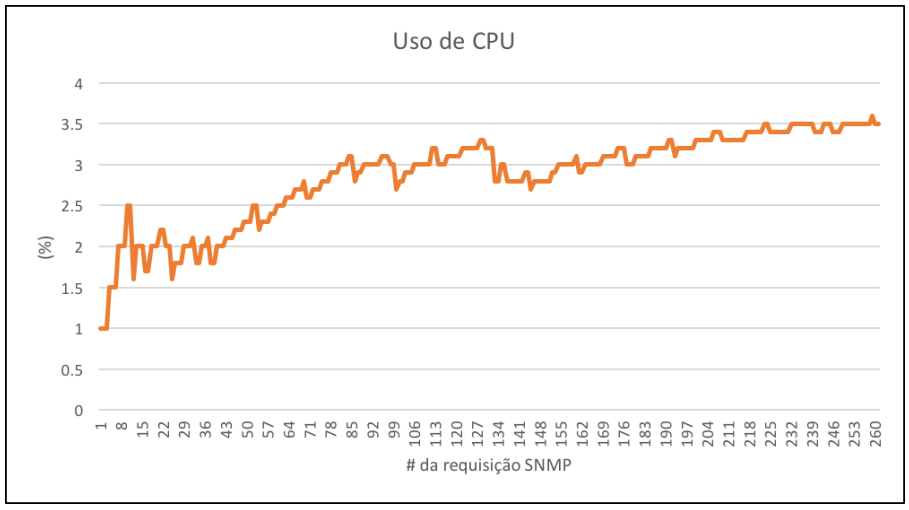


Figura 8. Utilização de CPU no *Gateway*

Com relação ao uso de memória, observou-se que o agente SNMP consome cerca de 4 MB da memória do *Gateway* (Raspberry Pi), o que corresponde a cerca de 0,5% da memória do mesmo sem variação significativa. Dessa forma, pode-se concluir que o *Gateway* é capaz de atender as requisições do módulo Gerente sem prejuízo de desempenho.

A partir dos resultados obtidos é possível verificar que a comunicação entre o *Gateway* e os dispositivos não exige grandes quantidades de recursos computacionais, podendo ser feita com dispositivos de baixo custo. O tempo de resposta total superior a 1 segundo para os três dispositivos pode ser explicado pela grande quantidade de variáveis (14) que são recuperadas em cada requisição. É possível otimizar essa comunicação para recuperar um subconjunto dos objetos da ENTITY-SENSOR-MIB, como, por exemplo, os valores dos sensores (*entPhySensorValue*) que mudam com frequência maior. No en-

tanto, a avaliação feita neste trabalho procurou explorar o pior caso, isto é, recuperando todas as variáveis de todos os dispositivos gerenciados.

6. Conclusões e Trabalhos Futuros

O monitoramento de objetos e a gerência de dispositivos representam um passo vital para a manutenção e controle da operação de um sistema em rede. Especificamente no contexto de ambientes IoT, poucas soluções existentes se preocuparam em com a padronização de tal monitoramento, especialmente para utilização em ambientes de *hardware* heterogêneo. Para abordar este problema, propôs-se um conjunto de adaptações na ENTITY-SENSOR-MIB, com o objetivo de padronizar o monitoramento de objetos de ambientes IoT.

Foi proposta uma arquitetura com múltiplos *Gateways*, onde mais de um *Gateway* pode gerenciar o mesmo dispositivo. Desta forma, espera-se que mesmo que um *Gateway* fique indisponível, outro (desde que também esteja monitorando o dispositivo em questão) possa manter a gerência e o controle sobre o dispositivo. Além disso, foi proposto um sistema de monitoramento, intitulado SAMURAI, para viabilizar os experimentos e estudos de caso da arquitetura proposta.

Na avaliação realizada, pôde-se observar que o tempo de resposta dos dispositivos ficou na casa dos 180 milissegundos por objeto consultado, demonstrando que a solução está dentro do esperado. Ainda, pode ser observado que mesmo com mais de 250 requisições SNMP, a utilização da CPU ficou abaixo de 4%, valor que representa pouco impacto para as demais funcionalidade do *Gateway*. Adicionalmente, o agente SNMP utiliza pouca memória do *Gateway*, ficando próximo de 0,5%. Por fim, o Sistema SAMURAI, ao consultar o *Gateway*, necessita aguardar pouco mais de 1 (um) segundo para obter as informações de todos os sensores monitorados, resultado importante, especialmente considerando a grande quantidade de objetos recuperados em mais de 250 requisições.

Como trabalhos futuros, pretende-se (*i*) investigar estratégias para garantir níveis de segurança, especialmente nas Camadas de *Troca de Dados* e *Integração de Informação*, (*ii*) realizar experimentos com diferentes dispositivos, simulando ambientes heterogêneos, (*iii*) avaliar a arquitetura e o sistema proposta em cenário com milhares de dispositivos para avaliar a escalabilidade da solução e (*iv*) comparar a solução proposta com outros protocolos de comunicação projetados para ambientes IoT como o MQTT e o CoAP.

Referências

Arduino (2016). URL: https://www.arduino.cc. Acessado em: abril de 2016.

Atzori, L., Iera, A., and Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, 54(15):2787 – 2805.

Benamar, N., Jara, A., Ladid, L., and Ouadghiri, D. E. (2014). Challenges of the Internet of Things: IPv6 and Network Management. In *2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 328–333.

Bierman, A., Romascanu, D., and Norseth, K. (2002). RFC 3433: Entity Sensor Management Information Base.

Bootstrap (2016). URL: http://getbootstrap.com. Acessado em: agosto de 2016.

Case, J. D., Fedor, M., Schoffstall, M. L., and Davin, J. (1990). RFC 1157: Simple network management protocol (SNMP).

Chang, K. D., Chen, J. L., Chen, C. Y., and Chao, H. C. (2012). IoT operations management and traffic analysis for Future Internet. In *2012 Computing, Communications and Applications Conference*, pages 138–142.

Chen, T. Y., Wei, H. W., Hsu, N. I., and Shih, W. K. (2013). A IoT Application of Safe Building in IPv6 Network Environment. In *IEEE COMPSAC 2013*, pages 748–753.

Cook, D. J. and Das, S. K. (2007). How smart are our environments? An updated look at the state of the art. *Pervasive and Mobile Computing*, 3(2):53 – 73.

Google (2016). URL: https://developers.google.com/chart/. Acessado em: agosto de 2016.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660.

Harrington, D., Wijnen, B., and Presuhn, R. (2002). RFC 3411: An architecture for describing simple network management protocol (SNMP) management frameworks.

Ma, H., Liu, L., Zhou, A., and Zhao, D. (2016). On Networking of Internet of Things: Explorations and Challenges. *IEEE Internet of Things Journal*, 3(4):441–452.

Montenegro, G., Kushalnagar, N., Hui, J., and Culler, D. (2007). RFC 4944: Transmission of IPv6 packets over IEEE 802.15. 4 networks.

NET-SNMP (2016). URL: http://net-snmp.sourceforge.net. Acessado em: abril de 2016.

Qin, Z., Denker, G., Giannelli, C., Bellavista, P., and Venkatasubramanian, N. (2014). A Software Defined Networking architecture for the Internet-of-Things. In *IEEE NOMS 2014*, pages 1–9.

Raspberry Pi (2016). URL: https://www.raspberrypi.org. Acessado em: agosto de 2016.

Ruby (2016). URL: https://www.ruby-lang.org. Acessado em: agosto de 2016.

Ruby Daemons (2016). URL: http://daemons.rubyforge.org. Acessado em: agosto de 2016.

Ruby on Rails (2016). URL: http://rubyonrails.org. Acessado em: agosto de 2016.

Santos, B. P., Silva, L. A. M., Celes, C. S. F. S., Neto, J. B. B., Peres, B. S., Vieira, M. A. M., Vieira, L. F. M., Goussevskaia, O. N., and Loureiro, A. A. F. (2016). *Livro dos Minicursos do SBRC 2016*, pages 1–50. Sociedade Brasileira de Computação.

Savolainen, T., Soininen, J., and Silverajan, B. (2013). IPv6 Addressing Strategies for IoT. *IEEE Sensors Journal*, 13(10):3511–3519.

Sehgal, A., Perelman, V., Kuryla, S., and Schonwalder, J. (2012). Management of resource constrained devices in the internet of things. *IEEE Communications Magazine*, 50(12):144–149.

Shelby, Z., Hartke, K., and Bormann, C. (2014). RFC 7252: The constrained application protocol (CoAP).

Trilha Principal do SBRC 2017
Sessão Técnica 11
Computação nas Nuvens II

Provisionamento Automático de Recursos em Nuvem IaaS: eficiência e limitações de abordagens reativas

Fábio Morais[1], **Raquel Lopes**[1], **Francisco Brasileiro**[1]

[1]Universidade Federal de Campina Grande (UFCG)
Laboratório de Sistemas Distribuídos, Campina Grande – PB – Brasil

fabio@lsd.ufcg.edu.br
{raquel,fubica}@dsc.ufcg.edu.br

Abstract. *Public cloud providers such as Amazon AWS and Rackspace offer programmable reactive auto-scaling services. This service addresses long running applications with time varying demands and works as follows: the client configures thresholds related to some application metric and when these thresholds are reached, actions are triggered in order to add or release resources from this application. For instance, we can configure the action of adding 2 new VMs whenever the average CPU utilization of the VMs running the application is greater than 70%. In this paper we perform an in-depth analysis of this type os reactive auto-scaling service, identifying its efficiency and limitations.*

Resumo. *Provedores públicos de Computação na Nuvem como Amazon AWS e Rackspace oferecem serviços programáveis de provisionamento automático e reativo de recursos. Esse serviços tratam de aplicações, de longa duração com demandas que variam no tempo, da seguinte forma: o cliente configura limiares relacionados a alguma métrica da aplicação e quando esses limiares são atingidos, ações são disparadas para adicionar ou liberar recursos da aplicação. Por exemplo, é possível configurar a ação de adicionar 2 novas VMs sempre que a utilização média de CPU das VMs executando a aplicação for maior que 70%. Nesse artigo realizamos uma análise aprofundada sobre esse tipo de serviço de provisionamento automático e reativo, identificando eficiências e limitações.*

1. Introdução

Provedores de Computação na Nuvem que oferecem infraestrutura como serviço (IaaS) são ambientes adequados para executar aplicações horizontalmente escaláveis. Normalmente, esse tipo de aplicação possui carga de trabalho variável no tempo e seu desempenho pode ser ajustado alterando o número de nós computacionais alocados para executá-la. As duas principais características de IaaS que incentivam a execução destas aplicações nestes ambientes são: (i) elasticidade da infraestrutura de execução; e (ii) modelo de tarifação em que se paga conforme a utilização do serviço (*pay-as-you-go*), permitindo que os provedores de aplicações paguem apenas pelos recursos adquiridos. Na prática, esses recursos são oferecidos no formato de máquinas virtuais (VMs) adquiridas pelos provedores das aplicações que são os usuários dos provedores IaaS.

O provisionamento automático de uma aplicação horizontalmente escalável consiste no ato de decidir periodicamente a quantidade de VMs necessárias para executar a aplicação no próximo intervalo de tempo de curto prazo (na ordem de minutos). Isso é realizado com o intuito de alcançar a qualidade de serviço (QoS) desejada para a aplicação,

ao mesmo tempo que os custos de infraestrutura são minimizados. Apesar da flexibilidade fornecida por ambientes de IaaS, o provisionamento automático ideal de uma aplicação desse tipo não é uma tarefa trivial. Primeiramente, é preciso antecipar a capacidade mínima requerida pela aplicação em um futuro próximo. Em seguida, decidir a quantidade de VMs necessária para atender a demanda da aplicação. Finalmente, essas decisões precisam ser continuamente realizadas para lidar com a variabilidade da carga de trabalho da aplicação ao longo do tempo.

Considerando um cenário em que o *provisionamento automático de recursos é oferecido como um serviço de IaaS*, a solução de provisionamento deve ser responsável por gerenciar a infraestrutura de execução da aplicação visando que esta atinja a QoS desejada usando a menor quantidade de recursos possível (*trade-off* custo/desempenho). Um outro aspecto importante deste serviço trata de questões de privacidade; espera-se que esse serviço utilize informações não-específicas da aplicação que podem ser coletadas no nível da infraestrutura de execução (como utilização de CPU, Memória, etc). Em geral, informações obtidas da aplicação (e.g. taxa de chegada de requisições, tipo de requisições, tamanhos de filas, etc) são consideradas sensíveis, não sendo facilmente compartilhadas.

O serviço de provisionamento segue um dos seguintes modos de operação: reativo ou proativo. O serviço proativo tenta antecipar a carga da aplicação para tomar decisões sobre a capacidade adequada. Já a técnica reativa, que é o foco deste artigo, consiste em uma reação programável às mudanças percebidas na aplicação e/ou na sua infraestrutura. Essencialmente, essa abordagem utiliza um conjunto de regras de provisionamento para decidir quando e em qual quantidade a aplicação deve ser provisionada [Lorido-Botrán et al. 2014]. O provisionamento reativo utiliza apenas informações sobre o estado atual da aplicação e seu ambiente para decidir sobre o provisionamento da aplicação.

Espera-se que os sistemas reativos não sejam eficientes ao prover aplicações com cargas de trabalho de intensa variabilidade no tempo [Lorido-Botrán et al. 2014]. Isso decorre da natureza reativa e pontual da solução e do fato da configuração das regras serem consideravelmente sensíveis a mudanças e tendências da carga de trabalho da aplicação, que geram a necessidade de ajustes frequentes mesmo na presença de um especialista na aplicação atuando no processo de configuração [Lorido-Botrán et al. 2014]. Desta forma, prováveis equívocos na configuração das regras podem provocar situações de sub-provisionamento que levam à degradação da QoS da aplicação e possíveis violações de SLO (*Service Level Objective*), ou de super-provisionamento, com o desperdício de recursos adquiridos do provedor de IaaS. Outro ponto negativo é que em geral as soluções reativas usam métricas intrusivas, específicas da aplicação, que não podem ser consideradas por um serviço de provisionamento genérico e automático em IaaS.

Apesar de criticada em termos de desempenho, a abordagem reativa ainda é muito explorada devido sua simplicidade e natureza intuitiva [Lorido-Botrán et al. 2014, da Rosa Righi et al. 2017]. Os principais provedores do mercado de Computação na Nuvem e IaaS, como Amazon Web Services (AWS) [Amazon 2016], Rackspace [Rackspace 2016] e Microsoft Azure [Azure 2016], oferecem serviços de provisionamento automático e reativo de recursos[1]. Esta abordagem também é amplamente explorada na

[1]A Google [Google 2016] oferece um serviço de provisionamento automático que também baseia-se na técnica reativa, mas utiliza um modelo de provisionamento mais sofisticado, não conhecido pelo público, para decidir a quantidade de VMs que devem ser provisionadas a cada intervalo de tempo.

literatura através do uso de diferentes conjuntos de métricas de desempenho, configurações de limiares e ações de provisionamento [Lim et al. 2009, Calheiros et al. 2012, Fito et al. 2010, Calcavecchia et al. 2012, Marshall et al. 2010, Bonvin et al. 2011, Ghanbari et al. 2011, Seung et al. 2011].

Neste artigo, avaliamos de forma profunda o desempenho de soluções reativas que utilizam apenas métricas de uso da infraestrutura e, portanto, podem ser implementadas como um serviço de provisionamento automático. O nosso objetivo é avaliar o desempenho de soluções reativas em termos de custo de execução e nível de QoS da aplicação provisionada, além de analisar o desempenho destas soluções em relação à aplicabilidade da técnica reativa no cenário de provisionamento automático como um serviço.

O restante deste artigo está estruturado da seguinte forma. A seguir a problemática do provisionamento automático como um serviço é descrito e detalhado (Seção 2). Na Seção 3 é realizado um levantamento sobre possíveis abordagens reativas do mercado e da literatura que podem ser empregadas como soluções de provisionamento automático em IaaS. Uma análise de desempenho de abordagens de provisionamento reativo segundo diferentes aspectos é realizada na Seção 4. Por fim, discute-se sobre os resultados obtidos (Seção 5) e conclui-se o trabalho com direcionamentos para futuros trabalhos (Seção 6).

2. Provisionamento Automático como um Serviço

O cenário de provisionamento automático como um serviço é composto essencialmente por quatro atores: (i) o provedor de IaaS; (ii) o provedor da aplicação a ser executada no ambiente de IaaS; (iii) o usuário final da aplicação provisionada que faz uso do serviço prestado pela mesma; e (iv) o provedor do serviço de provisionamento automático de recursos. A relação entre esses atores está representada na Figura 1.

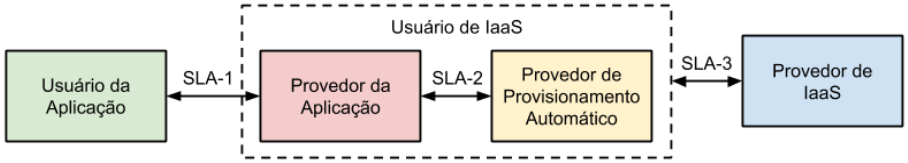


Figura 1. Visão geral da relação entre atores do serviço de provisionamento automático de recursos em ambientes de IaaS.

Tipicamente, o provedor de uma aplicação horizontalmente escalável adquire VMs de um provedor de IaaS para executar de forma dedicada a sua aplicação. Assim, o provedor da aplicação é usuário do serviço de IaaS, criando-se uma relação de serviço regida por um contrato de nível de serviço (SLA), identificado por SLA-3 na Figura 1. O provedor de IaaS garante (i) a aquisição e criação de VMs por parte de seu usuário e (ii) a disponibilidade de acesso a essas VMs, sob o risco de pagar penalidades. O serviço é tarifado segundo valores previamente definidos para cada intervalo de tempo de uso das VMs adquiridas e para os tipos de VMs adquiridas. Em um modelo público de IaaS, por exemplo, os provedores cobram um preço fixo por VM usada em um período menor ou igual ao ciclo de tarifação praticado pelo provedor (normalmente com duração de 1 hora).

A aplicação que executa no ambiente de IaaS é acessada remotamente pelos seus usuários e também deve haver um contrato que rege esse "serviço". Nesse caso o SLA-1 presente na Figura 1 indica o contrato estabelecido entre o provedor da aplicação e

os usuários da aplicação. Esses SLAs são responsáveis por garantir que os usuários da aplicação tenham acesso a um serviço com o nível de qualidade de serviço desejado.

O cenário deste estudo considera um quarto ator, que consiste em um serviço de provisionamento automático que é responsável por gerenciar os recursos alocados para executar a aplicação. Esse serviço atua como um procurador do responsável da aplicação junto ao provedor de IaaS, tendo o papel de adquirir e liberar recursos (tipicamente VMs) quando necessário, tornando-se também um usuário do serviço de IaaS.

O serviço de provisionamento automático realiza periodicamente o planejamento da capacidade da infra-estrutura (quantidade de VMs) para acomodar as flutuações da carga de trabalho da aplicação[2]. Para tal, esse serviço realiza as ações de provisionamento necessárias para cumprir o planejamento realizado. Essas flutuações de carga são vistas pelo serviço de provisionamento como variações na utilização dos diferentes tipos de recursos (CPU, memória, etc) executando a aplicação. Diferentes tipos de instância de VM podem ser selecionados para executar a aplicação durante sua execução [Morais et al. 2016], mas assume-se nesse estudo que apenas um tipo de instância é usado.

Quando a estratégia de provisionamento falha, decidindo por uma capacidade menor do que a necessária, o resultado é a degradação da QoS da aplicação e, possivelmente, perdas econômicas para o provedor da aplicação devido ao descumprimento do SLA-1. Isto acontece uma vez que os recursos ficam sobre-utilizados. O contrato entre o provedor da aplicação e do serviço de provisionamento automático (SLA-2 na Figura 1) deve considerar limiares adequados de uso dos recursos que levam à QoS desejada da aplicação. Pois quando esses limiares são atingidos, quanto maior for a utilização de um recurso menor será a QoS da aplicação, já que haverá mais competição pelo uso do recurso. Esses limiares definidos no SLA-2, quando descumpridos, conduzem a uma situação de saturação e queda da QoS da aplicação mencionada anteriormente.

Os SLAs entre os provedores das aplicações e o serviço de provisionamento (SLA-2) definem SLOs por meio de limiares de utilização dos recursos em uso para executar a aplicação. Por exemplo, um SLO pode definir que a utilização de CPU das VMs executando a aplicação não deve ultrapassar 80%. Assim, violações dos SLOs definidos no SLA-2 podem gerar degradação da QoS da aplicação, que é possivelmente percebida pelo usuário final. Se a utilização de recursos for mantida abaixo, mas o mais próximo possível do limiar estabelecido no SLO, os SLAs da aplicação (SLA-1) são satisfeitos. Além disso, quanto mais próximo dos limiares estabelecidos no SLA-2 estiverem as utilizações dos recursos que executam a aplicação, menor será o custo de execução da aplicação. Assim, o objetivo do serviço de provisionamento é manter os recursos alocados com utilizações mais próximas possíveis porém menores que o estabelecido nos SLOs do SLA-2.

É importante ressaltar que o serviço de provisionamento considerado deve ser capaz de prover recursos automaticamente para diferentes aplicações com o mínimo grau de intrusividade. Por isso, ele usa informações não específicas da aplicação. Além disso, o serviço deve buscar garantir a QoS desejada para a aplicação ao mesmo tempo que reduz o custo de execução da aplicação quando há variação em sua carga de trabalho.

[2]Para que este serviço de provisionamento automático seja possível, considera-se um sistema de monitoramento que coleta periodicamente a utilização dos recursos das VMs ativas que executam a aplicação.

3. Trabalhos Relacionados

O provisionamento reativo consiste na principal técnica de provisionamento utilizada pelo mercado de Computação na Nuvem e está majoritariamente presente nas soluções comerciais de IaaS [Amazon 2016, Rackspace 2016, Azure 2016]. A técnica reativa também é amplamente abordada pela literatura através do provisionamento automático baseado em informações específicas da aplicação (por exemplo, tempos de resposta, taxa de chegada de requisições, tipos de requisições, etc) [Calheiros et al. 2012, Fito et al. 2010, Calcavecchia et al. 2012, Marshall et al. 2010, Bonvin et al. 2011, Seung et al. 2011], que possuem uma relação direta com a QoS da aplicação. Acreditamos que um serviço de provisionamento automático precisa lidar apenas com métricas não-intrusivas obtidas, no nível da infraestrutura virtual, o que inviabiliza o uso dessas soluções para este fim.

As soluções reativas não-intrusivas que conhecemos [Ghanbari et al. 2011, Lim et al. 2009] assumem que a técnica simplesmente baseada em limiares inicialmente definidos não é suficiente para assegurar os objetivos de provisionamento, apesar de não haver nenhuma avaliação mais consistente deste fato. Por esse motivo, essas soluções combinam o uso de regras de provisionamento com outras técnicas de tomada de decisão e atualização de limiares. Ghanbari et al. combinam a técnica de regras reativas a um processo de voto, em que as VMs provisionadas devem concordar majoritariamente sobre as ações de provisionamento a serem realizadas. A solução de Lim et al. utiliza limiares proporcionais que são dinamicamente modificados para reduzir o impacto do provisionamento sob a utilização de recursos da infraestrutura e atenuar as limitações do uso de regras estaticamente definidas. Com objetivo semelhante, o trabalho de Netto et al. propõe o ajuste dinâmico da quantidade de VMs provisionadas por ação de provisionamento realizada [Netto et al. 2014].

Todavia, até onde se sabe não existe um estudo aprofundado sobre o desempenho de abordagens reativas de provisionamento automático a partir de métricas não-intrusivas, apesar da difusão dessa técnica no mercado de Nuvem e IaaS. Nesse sentido, este trabalho tem como objetivo analisar criteriosamente a eficiência e as limitações do uso dessa abordagem em um cenário de provisionamento automático como um serviço de IaaS.

4. Análise de Desempenho de Abordagens Reativas

Na prática, a abordagem reativa é a principal técnica de provisionamento atualmente em uso no mercado de Computação na Nuvem. Essa abordagem atua reagindo a mudanças na carga de trabalho da aplicação, que podem ser percebidas através das métricas monitoradas da aplicação ou da infraestrutura. As regras de provisionamento definem ações a serem tomadas em reação a mudanças. Em um cenário de provisionamento não-intrusivo, essas regras fazem uso dos limiares (*threshold-based*) de utilização da infraestrutura de execução (por exemplo, percentual de utilização de CPU) definidos no SLA-2 (Seção 2).

A configuração de cada regra é composta de duas partes essenciais. A primeira parte define a condição para disparo de uma ação de provisionamento. Esta parte é composta por um conjunto de triplas $\langle$métrica, limiar, operador condicional$\rangle$ que definem as condições para o disparo de ações de provisionamento. Por exemplo, uma ação é disparada se a métrica de utilização média de CPU for maior igual ao limiar de 70%. A segunda parte consiste na ação de provisionamento associada à condição de provisionamento definida; por exemplo, aumento da capacidade computacional da infraestrutura de

execução. Assim, quando uma condição de provisionamento for satisfeita (primeira parte) então uma ação de provisionamento associada (segunda parte) será disparada.

Pelo menos duas regras devem ser definidas: uma para adicionar recursos à infraestrutura de execução quando necessário e outra para remover recursos que não estão mais em uso. Em um ambiente de IaaS uma ação de provisionamento consiste na definição da quantidade fixa de VMs que deve ser adicionada ou removida da infraestrutura que executa a aplicação caso uma condição de disparo de ação seja satisfeita. Esse tipo de solução funciona como modelos de laço de controle que periodicamente avaliam se alguma condição de provisionamento foi satisfeita, e em caso positivo a ação associada é executada. Isso permite que em uma ambiente de IaaS a infraestrutura de execução possa ser dinâmica e automaticamente ajustada a partir das regras previamente definidas pelo responsável da aplicação, que também é um usuário da solução de provisionamento.

Nesse artigo, avaliamos o provisionamento automático e reativo como um serviço segundo os seguintes aspectos: (i) capacidade de manter a QoS da aplicação no nível desejado, idealmente não violando os SLOs da aplicação; (ii) capacidade de provisionar a aplicação com custos de execução próximos aos praticados por um serviço de provisionamento perfeito, livre de violações de SLO e com custo de execução mínimo; (iii) grau de generalidade em relação às aplicações provisionadas e a independência de características da carga de trabalho destas; e (iv) grau de controle permitindo que configurações diferentes lidem com o *trade-off* custo/QoS. Quanto maior o nível de QoS desejado, maior deve ser o custo e vice versa. É importante que o serviço ofereça esse controle ao seu usuário.

4.1. Modelo e dados de simulação

O modelo de simulação foi implementado na linguagem de programação R[3] [Chambers 2016]. Este modelo simula a interação entre o serviço de provisionamento automático, a aplicação provisionada, a infraestrutura virtual adquirida do provedor de IaaS e os componentes de monitoramento e atuação disponibilizados pelos provedores de IaaS. Dado o viés não intrusivo da solução de provisionamento, o componente de provisionamento opera desassociadamente da aplicação provisionada, de tal forma que as interações com as aplicações restringem-se à alocação ou liberação de VMs e à coleta de dados de utilização de recursos no nível da infraestrutura virtual alocada.

Um rastro de utilização consiste em uma série temporal da utilização de recursos de uma aplicação, onde cada item da série corresponde à média de utilização de CPU para o intervalo de tempo considerado. Cada item no rastro consiste em uma dupla $\langle x, y \rangle$, onde x é a utilização média de CPU no intervalo de tempo considerado ($x \in \mathbb{R}, 0 \leq x \leq 1$) e y é a quantidade de núcleos de CPU alocados à aplicação durante esse tempo ($y \in \mathbb{R}^+$). A demanda média da aplicação (u, $u \in \mathbb{R}^+$), em núcleos de CPU, é calculada com base nos dados de utilização média e alocação de CPU, dada por $u = x \times y$.

Nas simulações, a capacidade da infraestrutura alocada para executar a aplicação em número de núcleos de CPU é equivalente a quantidade de VMs alocadas, uma vez que o modelo de simulação considera VMs com apenas 1 núcleo de CPU. Desta forma, a utilização real simulada de cada máquina virtual entre o intervalo de tempo τ e $\tau + 1$ é computada como sendo o mínimo entre 100% e $\frac{u}{a}$, onde a é o número de VMs alocadas

[3]O código fonte do simulador encontra-se disponível em https://goo.gl/8a39zi

para executar a aplicação no experimento de simulação. Logo, cada VM alocada apresenta a mesma média de utilização para um mesmo intervalo de tempo τ. Assim, se a utilização de CPU no intervalo de tempo τ é maior do que o limite de utilização de CPU definido no SLO da aplicação, então ocorre uma violação de SLO, indicando que a capacidade a de CPU atribuída à aplicação não foi suficiente. O sistema de provisionamento opera com periodicidade configurável, de forma que a cada laço de controle pelo menos um novo item de dado de utilização de CPU é lido do rastro e esse dado é usado para computar a utilização real de CPU das VMs alocadas para o próximo intervalo de tempo.

O serviço de provisionamento simulado realiza o planejamento de capacidade da infraestrutura de execução para o próximo intervalo de tempo com base em regras de provisionamento reativo predefinidas. Desta forma a alocação de recursos ocorre em conformidade com as demandas simuladas da aplicação que são identificadas e supridas a partir da técnica de provisionamento. Finalmente, ao realizar a liberação de recursos da infraestrutura o simulador considera o modelo de IaaS operado pela Amazon, em que as VMs são tarifadas por hora. Assim, mesmo que o sistema de provisionamento decida pela desalocação de VMs em um dado momento, o desligamento de uma VM só é de fato efetuado se essa decisão ocorrer em sincronia com horas completas de uso dessa VM.

As simulações são alimentadas com rastros de utilização de CPU de 30 aplicações reais com duração média de 8 meses. Cada rastro consiste na medição da média de utilização de recursos de CPU produzida por uma única aplicação, monitorada a cada intervalo de 5 minutos. No período em que esses rastros foram capturados as aplicações estavam superprovidas de recursos de forma estática (a mesma infraestrutura de tamanho fixo executou a aplicação). Esses dados são provenientes de uma parceria realizada com a HP para o desenvolvimento de soluções de provisionamento automático de recursos em ambientes de IaaS[4] [Morais et al. 2013]. A função de distribuição acumulada (FDA) da utilização de CPU dessas aplicações é apresentada na Figura 2. A partir desta, percebe-se que os dados apresentam uma ampla variedade de distribuições de utilização entre as aplicações, com diferentes perfis de consumo de CPU. Além do mais, para uma mesma aplicação existe uma considerável variabilidade de intensidades de utilização de CPU, onde pelo menos metade das aplicações apresentam um desvio padrão da utilização maior que 15%. Dessa forma, considera-se que os dados utilização considerados são representativos para o estudo do provisionamento automático e reativo.

4.2. Provisionamento reativo perfeito

Inicialmente, analisamos a viabilidade de uma solução de provisionamento reativo que atue de forma perfeita, sem erros de provisionamento durante toda a execução da aplicação. Para tal, faz-se necessário que o serviço de provisionamento possua informações exatas sobre as demandas futuras de CPU da aplicação a cada ciclo de controle (intervalo de 5 minutos) para que o planejador de capacidade perfeito decida e aloque a quantidade necessária de VMs (com 1 núcleo de CPU), para assegurar os SLOs definidos para a aplicação com o mínimo custo. Um limiar de utilização de CPU de 100% foi considerado no SLO do serviço de provisionamento (SLA-2). Desta forma, a simulação resulta em um rastro da quantidade ideal de VMs alocadas no tempo para cada aplicação provisionada, assegurando custo mínimo e ausência de violações de SLO.

[4]Infelizmente, devido a questões de confidencialidade esses dados não encontram-se publicados.

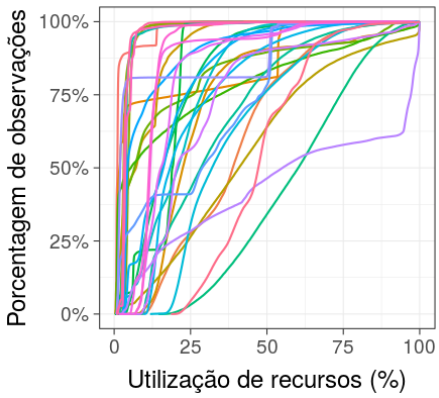


Figura 2. FDA da utilização de CPU das aplicações consideradas.

A partir desses rastros foi realizada uma análise *post-mortem* para identificar a partir das ações de provisionamento realizadas, quais seriam as regras de provisionamento reativo mais apropriadas. Consideramos que cada ação de provisionamento necessita de no máximo 5 minutos para ser efetivada, tempo referente a capacidade de reação da técnica de provisionamento e em conformidade com a mínima periodicidade disponibilizada pelos dados de simulação. Assim, para uma ação de provisionamento ocorrida no tempo τ a configuração de provisionamento necessária para realizar essa ação é computada com base em dados obtidos no tempo $t - 2$. Por exemplo, se no intervalo de tempo τ, 2 VMs foram acrescentadas pelo serviço perfeito, então a análise *post-mortem* identifica qual seria a regra de provisionamento disparada em $t - 2$ que resultaria nessa mesma decisão.

Com base nas regras descobertas nós verificamos a viabilidade de se configurar um serviço reativo de provisionamento que seja próximo ao ótimo. Isto é possível se as mesmas regras (ou regras semelhantes) se repetem para uma aplicação e, idealmente, para várias aplicações. Não foi esse o resultado encontrado, como veremos nas seções a seguir.

4.2.1. Não há predominância de configuração de limiares

Com base nas configurações inferidas do processo de provisionamento perfeito identificamos todas as diferentes regras necessárias para gerar o provisionamento perfeito. A fim de reduzir o espaço de possibilidades de configuração, os valores dos limiares inferidos a partir do provisionamento perfeito foram sumariados em faixas de valores com tamanho de 5% (por exemplo, um limiar de 31% passa a pertencer a faixa de 35%). Desta forma, os limiares originalmente inferidos foram categorizados em 20 grupos de limiares, com valores entre 5% e 100%, em passos de 5%.

Computamos a frequência de diferentes limiares nas regras de provisionamento de cada uma das aplicações, agrupados por ação de provisionamento. A Figura 3 (à esquerda) mostra o diagrama de caixa da frequência de uso do limiar mais frequente no provisionamento de cada aplicação. Observa-se que para 50% das aplicações provisionadas a mediana de frequência dos limiares mais predominantes foi de apenas aproximadamente 17%, para ambos os tipos de ação de provisionamento. Além da não predominância de um limiar por aplicação, também observa-se uma quantidade significativa de diferentes configurações de limiares que são usadas por aplicação. Em média, 17 diferentes confi-

gurações são necessárias por aplicação, como apresentado na Figura 3 (à direita).

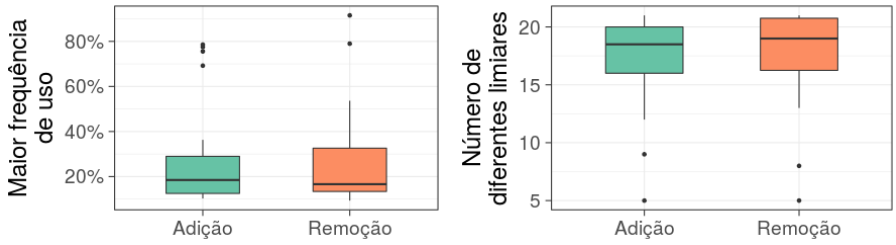


Figura 3. Análise por aplicação da frequência de uso do limiar mais comum (à esq.) e do número de diferentes configurações de limiares por aplicação (à dir.).

Desta forma, no que se refere à predominância de configuração no provisionamento reativo perfeito, a frequência de uso de configurações de limiares e da quantidade de diferentes configurações por aplicação observadas evidenciam a dificuldade de uso da abordagem reativa para gerar um cenário de provisionamento perfeito dessas aplicações. Em adição à não predominância de regras específicas, regras incompatíveis foram muitas vezes identificadas, como por exemplo, o mesmo limiar era associado ora à ação de remoção, ora à ação de adição.

4.2.2. Não há padrão em transições de limiares usados consecutivamente

Um outro aspecto sobre a configuração de limiares para o provisionamento perfeito dessas aplicações consiste na variabilidade temporal dos limiares das regras aplicadas no provisionamento. Além da necessidade de decidir quais as configurações de limiares devem ser usadas para cada aplicação, também deve-se conhecer como ocorrem as mudanças de uma configuração para outra diferente ao longo do provisionamento. A Figura 4 (à esquerda) apresenta o percentual de ocorrência de transições entre limiares de utilização diferentes, ou seja a frequência de mudança de uso de limiares entre intervalos de provisionamento perfeito. Para metade das aplicações, em 77% das ações de provisionamento consecutivas ocorrem mudanças no limiar da regra de provisionamento utilizada.

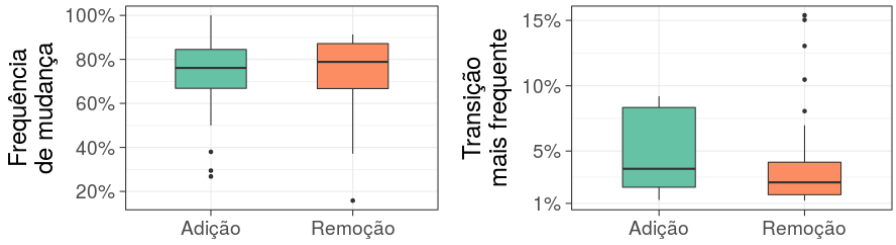


Figura 4. Análise da frequência de transições entre diferentes limiares de utilização (à esq.) e da frequência da transição mais recorrente no provisionamento por aplicação (à dir.)

Uma outra questão que buscamos responder diz respeito a padrões de transição de um limiar para outro usados em sequência. Verificamos que a frequência de recorrência de uma mesma transição entre regras não é considerada significativa. Como pode ser visto

na Figura 4 (à direita), para todos os cenários, em 90% das mudanças de limiares mais recorrentes por aplicação a frequência de observação dessas transições é menor que 9% do total de transições realizadas. Desta forma, observa-se que o percentual de ocorrência de uma mesma transição entre limiares também desfavorece a eficiência de configuração de uma solução perfeita de provisionamento reativo baseado em utilização de CPU.

4.2.3. Existe uma relação forte entre carga de trabalho e ações de provisionamento

Realizamos uma análise para entender a relação entre a variação da utilização de CPU medida na infraestrutura de execução e a quantidade de operações de provisionamento necessárias no provisionamento perfeito. Calculamos de forma pareada a correlação entre o desvio padrão do consumo de CPU por aplicação e a quantidade de operações de provisionamento realizadas no provisionamento reativo perfeito de cada aplicação. Os coeficientes de Spearman $(0, 91)$ e Kendall $(0, 75)$ mostram que essa relação é de forte a muito forte para as aplicações consideradas. Ou seja, quanto maior for a variabilidade da carga de trabalho da aplicação maior será a necessidade de realizar operações de provisionamento para a abordagem perfeita.

Além disso, também existe uma correlação significativa entre a variabilidade de carga de trabalho e o número de diferentes configurações de limiares de provisionamento. Os coeficientes de correlação de Spearman e Kendall entre o desvio padrão do uso de CPU e a quantidade de diferentes limiares usados no provisionamento perfeito apresentam correlações com valor de $0, 61$ e de $0, 45$, respectivamente. Assim, assume-se que a configuração dos limiares de provisionamento são dependentes de características específicas da carga de trabalho das aplicações provisionadas. Esse resultado limita significativamente o desempenho da abordagem reativa em relação à eficiência e à capacidade de generalidade de configuração para um conjunto heterogêneo de aplicações.

4.3. Abordagens reativas na prática

Os objetivos de provisionamento automático de aplicações em ambientes de IaaS envolvem um *trade-off* entre a redução do custo de provisionamento e a redução do número de violações de SLO. Em um cenário de provisionamento perfeito esses objetivos conflitantes são otimizados, gerando uma execução da aplicação sem violações de SLO com o menor custo possível de execução. O controle desses objetivos pode ser realizado a partir da configuração dos limiares de provisionamento. Quanto maior for o limiar de provisionamento, seja ele de adição ou remoção, prioriza-se a redução de custos de execução e consequentemente aumenta-se a chance de violações de SLO. Por outro lado, quanto menores forem os limiares, maior será o custo de execução da aplicação, já que VMs são adicionadas ao primeiro sinal de aumento de utilização e só são removidas quando a utilização está suficientemente baixa. Por consequência, a probabilidade de ocorrência de violações nesse caso é menor.

Essa relação fica evidente ao analisarmos o provisionamento reativo considerando diferentes configurações de limiares. Simulamos cenários de provisionamento com limiares de adição de VMs variando de 20% a 90%, em passos de 10%, com limiares de remoção a uma distância absoluta do limiar de adição variando de 10% a 80%, também

em passos de 10% [5] e ações de provisionamento que correspondem a adição ou remoção de uma VM com um núcleo de CPU. A Figura 5 apresenta o resultado em termos de violações de SLO[6] e custo do serviço reativo com as diferentes configurações. No eixo X são apresentados os diversos limiares de adição usados e a cor de cada diagrama de caixa indica o limiar de remoção usado (ver os rótulos acima do gráfico). O *trade-off* fica evidente nos resultados: para os menores limiares de adição/remoção temos os maiores custos e consequentemente as menores violações de SLO. E o oposto também é visto, os maiores limiares de adição/remoção levam aos menores custos e maiores violações de SLO. É evidente que ao controlar esses limiares estamos controlando o quanto de violação aceitamos (ou o quanto de custo estamos dispostos a pagar).

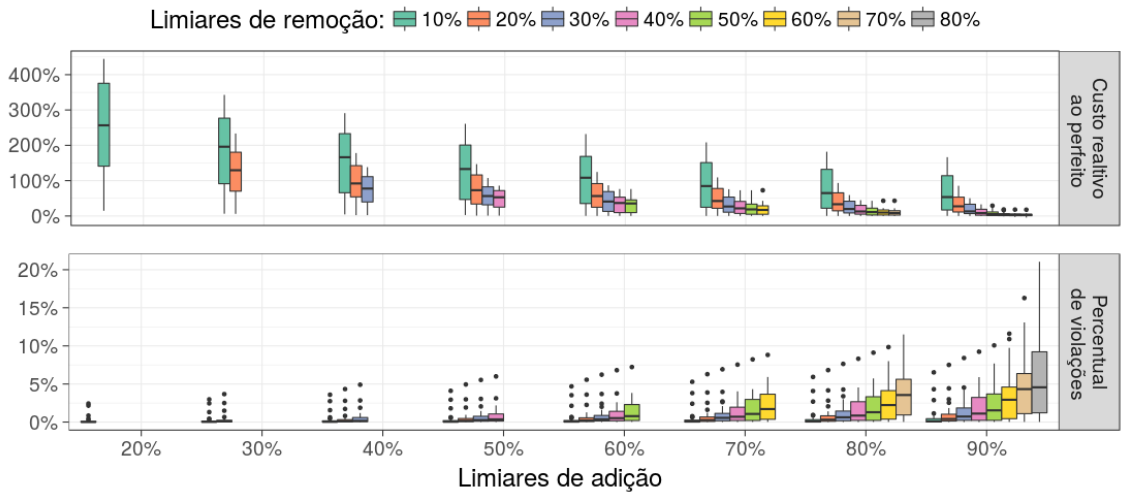


Figura 5. Desempenho da abordagem de provisionamento reativo em termos do custo de provisionamento e do percentual de violações de SLO.

Controlar esses limiares, no entanto, não é trivial, pois o que comumente buscamos é a manutenção da QoS desejada com o menor custo possível de provisionamento. Existem valores de limiares de adição/remoção que levam várias aplicações a poucas violações? Agrupamos os resultados conforme o percentual de violação de SLOs observado. Três classes foram definidas: $= 0\%$ (sem violações), $\leq 0,1\%$ e $\leq 1\%$. A partir da Figura 6 é possível identificar a quantidade de aplicações cujas violações de SLO enquadram-se nas diferentes classes considerando as diferentes configurações analisadas. Verificamos que nenhuma das configurações de provisionamento simuladas foi suficiente para atingir os objetivos de todas as aplicações, mesmo para o cenário mais flexível com um limite máximo de 1% de violações de SLO. Nesse cenário o percentual médio de aplicações cujo limite de violações foi respeitado é de cerca de 67%.

Restringindo o limite de violações para 0,1% ocorre um redução da capacidade da solução reativa em atingir o objetivo de QoS da aplicação, com uma média de sucesso para aproximadamente 30% das aplicações e de 60% para as configurações com melhor desempenho (barras mais a esquerda). Para o cenário mais conservador, em que violações de SLO não são aceitas, a meta só é atingida por 26% das aplicações, no melhor caso.

[5]Limiares de adição são sempre configurados com valores maiores que os dos limiares de remoção.
[6]Percentual de intervalos de controle em que a utilização de CPU atingiu 100%.

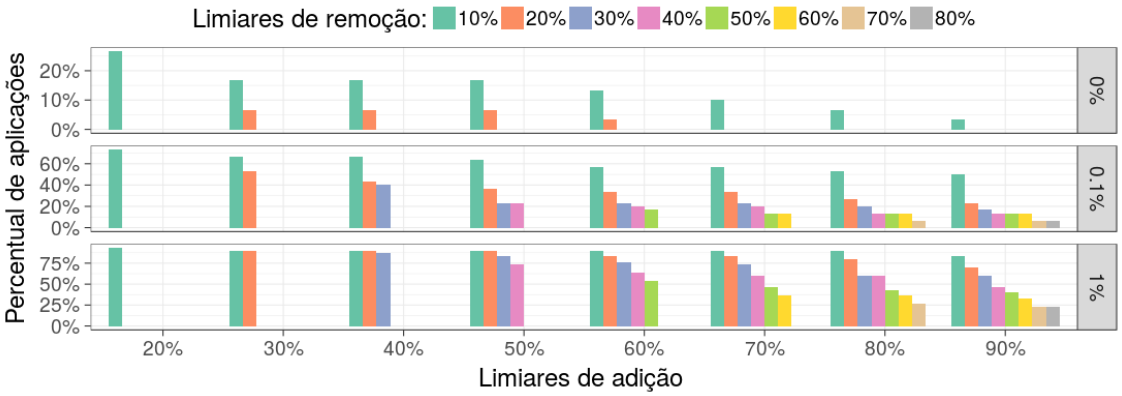


Figura 6. Análise do percentual de aplicações em que foi possível atingir os objetivos de QoS em termos dos limites de violações de SLO.

É importante lembrar que o cumprimento dos objetivos de QoS estão associados a custos de provisionamento, e quanto mais restritivo o limite aceitável de violações de SLO maior será o custo de provisionamento. Para as configurações mais conservadoras (menores limiares de adição/remoção) o custo pode chegar a ser até 400% maior que o custo alcançado pelo provisionamento perfeito. Na prática, um custo proibitivo. A avaliação do custo incorrido para cada um dos objetivos de SLO pode ser melhor observada na Figura 7. O diagrama de caixa mostra, para cada aplicação cujos objetivos de QoS foram satisfeitos, o menor custo de provisionamento relativo ao provisionamento perfeito, que consiste na seleção não realista do melhor cenário de configuração.

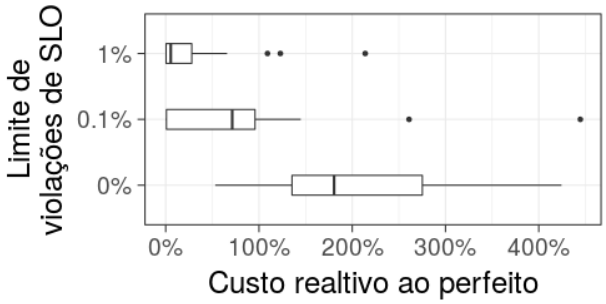


Figura 7. Análise de custo relativos ao provisionamento perfeito para diferentes cenários de limites de violações de SLO.

Para a classe com 0% de violações de SLO, metade das aplicações que atingiram esse objetivo apresentaram custo 180% maior que o custo do provisionamento perfeito correspondente. Para um limite máximo de $0,1\%$ de violações de SLO o custo médio é 71% maior que o obtido no provisionamento perfeito. Apenas no cenário mais flexível, em que o limite de violações de SLO é de 1%, o custo de provisionamento da abordagem reativa aproxima-se dos custos obtidos pela abordagem perfeita, com uma elevação de no máximo 6% de custo para metade das aplicações cujo objetivo foi atingido.

5. Discussão

A abordagem reativa não é adequada para a construção de um serviço de provisionamento no ambiente de IaaS. Várias são as razões que nos levam a essa conclusão. Primeiramente, a configuração das regras de reação é sensível às características das cargas de trabalho das aplicações, estando diretamente relacionadas com os perfis de consumo de CPU, o que contesta a generalidade e independência da solução em relação à aplicação provisionada. Um outro ponto que merece destaque é que a nossa análise não revelou uma configuração de limiar predominante para uma aplicação, muito menos para várias aplicações, o que inviabiliza o uso de uma configuração padrão. É senso comum acreditar que adicionar nós quando se chega em utilização de $\approx 70\%$ e remover nós quando se chega em $\approx 40\%$ é adequado, quando a nossa análise demonstra que não. Além disso, a configuração da quantidade de VMs provisionadas por ação disparada é em um outro fator, não tratado nesse trabalho, de ineficiência da abordagem. Finalmente, a abordagem reativa não é, na prática, bem sucedida em cumprir os objetivos de QoS das aplicações e, quando cumpre, leva a custos muito elevados. Em outras palavras, mesmo que soubéssemos como configurar o sistema (o que não sabemos, por isso uma varredura de parâmetros foi realizada), a abordagem reativa não foi bem sucedida em lidar com o *trade-off* custo/QoS.

6. Conclusões e Trabalhos Futuros

Neste trabalho foi realizada uma análise sobre a eficiência e limitações de abordagens reativas de provisionamento automático em IaaS. A análise foi realizada a partir da complexidade de configuração de regras de provisionamento perfeito e do desempenho de uma abordagem reativa prática em relação a violações de SLO e custo de provisionamento.

Foi demonstrado que a eficiência da técnica está relacionada com os padrões de carga de trabalho das aplicações, o que inviabiliza o seu uso como um serviço genérico de provisionamento. Além disso, mesmo para um conjunto com as configurações de provisionamento mais eficientes por aplicação, o desempenho da técnica mostra-se não eficaz em assegurar os objetivos mais restritivos de QoS da aplicação, mesmo com elevados custos de provisionamento em comparação a um cenário de provisionamento perfeito.

Como trabalhos futuros, pretende-se avaliar a construção de serviços de provisionamento automático a partir de abordagens de provisionamento mais sofisticadas, que busquem adaptar-se às variabilidades presentes nas cargas de trabalho. Além de uma avaliação da abordagem de provisionamento considerando diferentes métricas de utilização de recursos como base do provisionamento automático e reativo em ambientes de IaaS.

Agradecimentos

Essa pesquisa foi parcialmente financiada pela CAPES e pelo projeto EU-BRA BigSea (MCTI/RNP). Francisco Brasileiro é pesquisador do CNPq (processo 311297/2014-5).

Referências

Amazon (2016). Amazon auto scaling. `https://goo.gl/s1Zzx9`. Nov. 2016.

Azure, M. (2016). Autoscale a cloud service. `https://goo.gl/2CNo66`. Nov. 2016.

Bonvin, N., Papaioannou, T., and Aberer, K. (2011). Autonomic sla-driven provisioning for cloud applications. In *Cluster, Cloud and Grid Computing, 11th IEEE/ACM International Symposium on*, CCGRID '11, Newport Beach, CA, USA.

Calcavecchia, N., Caprarescu, B., Di Nitto, E., Dubois, D., and Petcu, D. (2012). Depas: a decentralized probabilistic algorithm for auto-scaling. *Computing*, 94:701–730.

Calheiros, R., Vecchiola, C., Karunamoorthy, D., and Buyya, R. (2012). The aneka platform and qos-driven resource provisioning for elastic applications on hybrid clouds. *Future Generation Computer Systems*.

Chambers, J. (2016). The R project for statistical computing. `https://goo.gl/NrCZaJ`. Nov. 2016.

da Rosa Righi, R., Rodrigues, V. F., Rostirolla, G., da Costa, C. A., Roloff, E., and Navaux, P. O. A. (2017). A lightweight plug-and-play elasticity service for self-organizing resource provisioning on parallel applications. *Future Generation Computer Systems*.

Fito, J., Goiri, I., and Guitart, J. (2010). Sla-driven elastic cloud hosting provider. In *Parallel, Distributed and Network-Based Processing, 18th Euromicro International Conference on*, PDP '10, Pisa, Italy.

Ghanbari, H., Simmons, B., Litoiu, M., and Iszlai, G. (2011). Exploring alternative approaches to implement an elasticity policy. In *Cloud Computing (CLOUD), IEEE International Conference on*.

Google (2016). Google cloud autoscaler. `https://goo.gl/LjITDz`. Nov. 2016.

Lim, H., Babu, S., Chase, J., and Parekh, S. (2009). Automated control in cloud computing: challenges and opportunities. In *Automated control for datacenters and clouds, 1st Workshop on*, ACDC '09, Barcelona, Spain.

Lorido-Botrán, T., Miguel-Alonso, J., and Lozano, J. A. (2014). A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of Grid Computing*.

Marshall, P., Keahey, K., and Freeman, T. (2010). Elastic site: Using clouds to elastically extend site resources. In *Cluster, Cloud and Grid Computing, 10th IEEE/ACM International Conference on*, CCGRID '10, Melbourne, Victoria, Australia.

Morais, F., Brasileiro, F., Lopes, R., Araújo, R., Satterfield, W., and Rosa, L. (2013). Autoflex: Service agnostic auto-scaling framework for iaas deployment models. In *Cluster, Cloud and Grid Computing, 13th IEEE/ACM International Symposium on*, CCGRID '13, Delft, Netherlands.

Morais, F., Lopes, R., and Brasileiro, F. (2016). Instance type selection in proactive horizontal auto-scaling. In *Cloud Computing Technology and Science, 8th IEEE International Conference on*, CloudCom '16, Luxembourg.

Netto, M. A., Cardonha, C., Cunha, R. L., and Assunçao, M. D. (2014). Evaluating auto-scaling strategies for cloud computing environments. In *Modelling, Analysis & Simulation of Computer and Telecommunication Systems, 22nd IEEE International Symposium on*, MASCOTS '14, pages 187–196. IEEE.

Rackspace (2016). Rackspace autoscale. `https://goo.gl/dxRR8Z`. Nov. 2016.

Seung, Y., Lam, T., Li, L., and Woo, T. (2011). Cloudflex: Seamless scaling of enterprise applications into the cloud. In *Computer Communications, 30th IEEE International Conference on*, INFOCON '11, Shanghai, China.

Coordenação de *Containers* no Kubernetes: Uma Abordagem Baseada em Serviço

Hylson Vescovi Netto[1,2], Caio Pereira Oliveira[1], Aldelir Fernando Luiz[2], Lau Cheuk Lung[1], Luciana de Oliveira Rech[1], José Roque Betiol Júnior[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina

[2]*Campus* Blumenau – Instituto Federal de Educação, Ciência e Tecnologia Catarinense

{hylson.vescovi,aldelir.luiz}@blumenau.ifc.edu.br,
{lau.lung,luciana.rech}@ufsc.br, {caiopoliveira,roque.betioljr}@gmail.com

Resumo. *Constantemente o paradigma de computação em nuvens tem sofrido modificações. Uma das principais consiste no uso da virtualização por contêineres, com vista para uma maior flexibilidade na infra-estrutura. A criação do gerenciador de contêineres denominado Kubernetes, por parte do Cloud Native Computing Foundation (CNCF), expressa os esforços para orientar tais modificações de maneira padronizada. O Kubernetes permite a replicação de contêineres, porém, não do estado das aplicações hospedadas. Neste sentido, este artigo versa sobre a replicação do estado de contêineres, em que é proposta uma arquitetura de sistema para efetuar a coordenação sob forma de serviço, a fim de acoplar-se à aplicação de maneira leve e simples. Como prova de conceito, foram realizados experimentos a fim de analisar o comportamento de aplicações com graus de consistência forte e eventual, cujos resultados demonstraram a viabilidade da proposta.*

Abstract. *The cloud computing paradigm have been modified. One of the most impacting changes is the usage of system level virtualization, for a better infrastructure management. The creation of Kubernetes, a containers management system, by the Cloud Native Computing Foundation (CNCF) express the efforts to guide the changes in a standard manner. Kubernetes can replicate containers, but not the state of the applications hosted in the containers. This paper presents an architecture for replicating state in containers providing coordination as a service, with a light and simple coupling to the application. Some experiments analyse the behavior of applications which observe eventual and strong consistency when reading data. The results shown that the proposal is feasible.*

1. Introdução

A profusão do paradigma de computação em nuvem (do inglês, *cloud computing* [Mell and Grance 2011]) no cotidiano das pessoas e organizações tem impelido quanto ao uso extensivo da tecnologia de virtualização [Popek and Goldberg 1974, Smith and Nair 2005], em face ao provisionamento dinâmico de recursos – um dos pilares fundamentais deste modelo de negócio –, o que portanto, vem de encontro à definição de nuvem computacional proposta pelo NIST [Mell and Grance 2011]. Um dos aspectos mais propulsores acerca da adoção da tecnologia de virtualização no âmbito das nuvens computacionais, decorre da permissibilidade de isolamento das cargas de trabalho executadas sobre tal ambiente, além da possibilidade da realização de algum controle sobre os recursos provisionados.

Embora a virtualização clássica (i.e., aquela onde o sistema de computação é virtualizado por completo) seja, indiscutivelmente, a tecnologia predominante no âmbito de provedores de nuvens computacionais (i.e., *data centers*), uma alternativa bastante atrativa na atualidade consiste na virtualização baseada em *containers* [Soltesz et al. 2007] – doravante referenciado por contêiner(es). A virtualização baseada em contêineres se caracterizada como uma tecnologia capaz de lançar instâncias de processamento isoladas umas das outras, isto é, numa mesma instância do sistema operacional, em que cada contêiner tem sua própria abstração de recursos. Deste modo, o isolamento não requer a execução das aplicações em separado, por diferentes instâncias de sistema operacional, tampouco o controle por algum monitor de máquinas virtuais, a exemplo do que ocorre com a virtualização clássica.

Dentre as tecnologias recentes para o suporte a virtualização por contêineres, uma das implementações mais populares na atualidade é o *Docker* [Bernstein 2014], a qual pode ser vista como uma extensão do LXC (i.e., *Linux Containers*) [Bernstein 2014]. Por outro lado, é digno de nota que a implementação nativa do *Docker* não provê mecanismos que possibilitam o gerenciamento/orquestração/integração dos contêineres num ambiente de *cluster* – um requisito desejável para o provimento de tolerância a faltas. Diante disso, alguns engenheiros do Borg [Verma et al. 2015] – o atual sistema de gerenciamento de contêineres do Google –, no âmbito do CNCF, trabalharam na proposição de um sistema denominado Kubernetes [Verma et al. 2015] – um sistema que visa o controle e gerenciamento do ciclo de vida de contêineres num ambiente de *cluster*.

Em suma, o Kubernetes replica os contêineres no intuito de melhorar a disponibilidade do ambiente virtualizado. Assim, aqueles contêineres que porventura apresentarem estado de falha serão recriados pelo Kubernetes, embora o estado da(s) aplicação(ões) em execução no(s) mesmo(s) não seja(m) recuperado(s). Porém, tal funcionalidade é passível de implementação a partir do uso de volumes externos. Todavia, é importante se ater ao fato de que o êxito na persistência do estado da aplicação só é obtido se os volumes puderem ter alguma proteção contra falhas, além de que a aplicação deverá exercer o controle sobre os acessos concorrentes ao volume em questão.

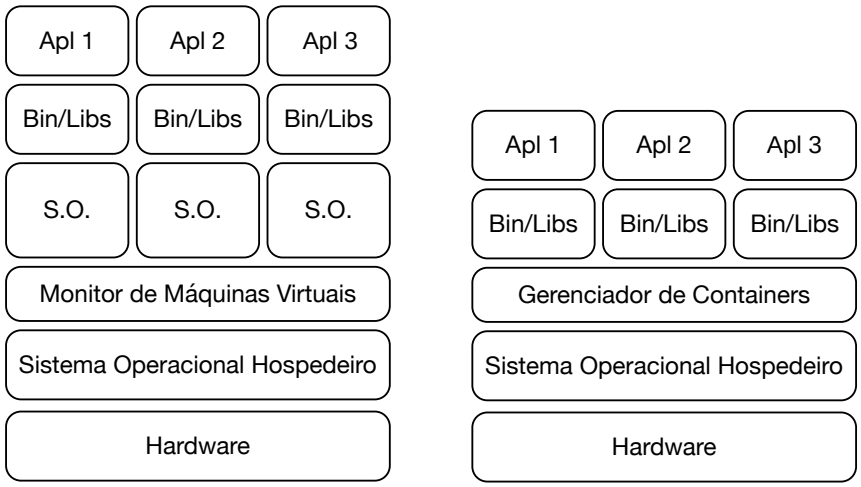
Diante do exposto, este trabalho visa a proposição de uma solução arquitetural para efetuar a coordenação da alteração de estados no Kubernetes, por meio de uma abordagem baseada em serviço. No que segue, o trabalho está organizado da seguinte maneira: a Seção 2 dedica-se a revisar a literatura acerca da tecnologia de virtualização baseada em contêineres; a Seção 3 descreve os detalhes de especificação empregados na proposta objeto deste trabalho, tais como arquitetura e algoritmos desenvolvidos; a Seção 4 apresenta os resultados experimentais obtidos a partir de um protótipo implementado; a Seção 5 estabelece relações com trabalhos da literatura, e por fim; na Seção 6 são evidenciadas as conclusões e alguns apontamentos de trabalhos futuros.

2. Tecnologia de Virtualização Baseada em Containers

A virtualização consiste numa tecnologia que permite abstrair os recursos físicos de um sistema de computação, tendo como propósito o provimento de um ambiente computacional virtual capaz de melhorar o aproveitamento dos recursos disponíveis [Parmelee et al. 1972, Popek and Goldberg 1974, Smith and Nair 2005]. De um modo geral, a virtualização ocorre por meio de uma máquina virtual, que é caracterizada como um ambiente de software construído sobre as interfaces fornecidas pelos recursos físicos disponíveis [Smith and Nair 2005]. Neste sentido, a virtualização em nível de sistema, doravante

denominada por virtualização baseada em contêineres, surgiu no contexto do sistema operacional FreeBSD, como uma versão estendida do comando *chroot*, denominada *jail* [Bernstein 2014]. Oportunamente, a SUN MicroSystems realizou modificações sobre o *jail*, incorporando-o ao Sistema Operacional Solaris, tendo denominado tal sistema por *zone*.

Em suma, a virtualização baseada em contêiner é similar às tecnologias de virtualização tradicionais (p. ex.: *VMWare*, *Xen*, etc.), no sentido de permitir que diversas aplicações – Apl n das Figuras 1(a) e 1(b) – sejam executadas de maneira isolada num mesmo sistema de computação (p. ex.: S.O. e hardware). Ao passo que a virtualização tradicional requer a execução de um sistema operacional completo (i.e., o convidado) num sistema hospedeiro para prover um ambiente isolado (cfm. a Figura 1(a)), um contêiner compartilha o núcleo subjacente do sistema hospedeiro e isola os processos executados nele, dos demais processos em execução no sistema hospedeiro (vide Figura 1(b)). Isto é, instâncias de contêineres virtualizam o próprio sistema operacional em vez de virtualizar o hardware, de modo que um gerenciador de contêiner faz o papel de mediador de uso e acesso aos recursos disponíveis no sistema de computação subjacente.



(a) Máquinas Virtuais (b) Containers

Figura 1. Modelos de arquitetura de tecnologias de virtualização.

A despeito das similaridades verificadas nas tecnologias de virtualização tradicional e baseada em contêineres, a segunda apresenta maior dinamicidade em termos de ciclo de vida dos processos/aplicações, uma vez que cabe a eles a simples tarefa de apenas iniciar ou destruir processos em seu espaço isolado. Outrossim, quando comparado às máquinas virtuais tradicionais os contêineres se destacam por sua eficiência, pois, não somente a sua instanciação, mas também o provisionamento dos recursos ocorre de maneira bastante rápida.

Não obstante, a virtualização por contêineres também é impulsionada pela portabilidade verificada em tal tecnologia, bem como pelo uso racional dos recursos – uma aplicação num contêiner só consome recursos quando é lançada uma instância do mesmo. Por outro lado, um contêiner é caracterizado como uma máquina virtual sem estado, uma vez que as imagens pelas quais instanciam-se os contêineres são estáticas. Assim, quando um contêiner é encerrado, não apenas seu estado, mas também o daquelas aplicações que nele estavam em execução são perdidos.

Embora os pontos ora elucidados apontem para vantagens quanto ao uso da tec-

nologia de contêineres, as implementações atuais de contêineres não dispõem de suporte para o gerenciamento de instâncias num ambiente de *cluster* (p. ex.: *docker*[1], *shifter*[2]), um requisito desejável se considerada a disponibilidade de aplicações executadas em contêineres. Neste sentido, no intuito de melhorar a disponibilidade de aplicações em contêineres o Google criou o Kubernetes [Bernstein 2014].

Mais precisamente, o Kubernetes consiste num ambiente gerenciador de contêineres, com a finalidade de administrar o ciclo de vida de contêineres em nós num ambiente de *cluster*. Dentre as funcionalidades providas pelo Kubernetes, pode-se citar como principais: controle de admissão dos contêineres, balanceamento de recursos, descoberta de serviços entre os contêineres, publicação do serviço para acessos externos ao *cluster* e o balanceamento de carga entre os contêineres.

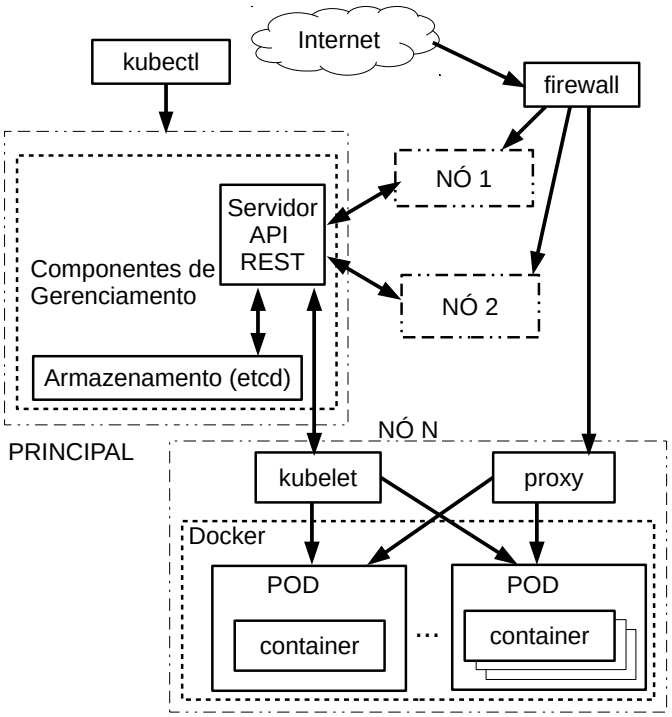


Figura 2. Arquitetura do Kubernetes

A Figura 2 descreve de maneira simplificada a arquitetura do Kubernetes. Note que o Kubernetes é composto por máquinas (virtuais ou físicas) denominadas nós. Os componentes POD consistem na unidade básica na qual o Kubernetes opera (i.e., uma aplicação, por exemplo), de modo que cada POD pode ter um ou mais contêineres. No contexto de um POD, os contêineres podem compartilhar recursos, por exemplo, um volume de dados externo. O *firewall* é responsável por despachar as requisições dos clientes para os nós presentes no *cluster* gerido pelo Kubernetes – cada requisição é entregue a apenas um nó.

O *proxy* tem a finalidade de encaminhar as requisições aos PODs e estes, por sua vez, podem ser replicados ou não. O Kubernetes pode replicar contêineres a fim de aumentar a disponibilidade de aplicações. Quando um contêiner falha, o Kubernetes recria o contêiner a partir de uma imagem. Assim, quando os PODs são replicados, as requisições são distribuídas pelo critério do algoritmo *Round Robin*. Por sua vez, os PODs são gerenciados pelo

[1]`https://github.com/docker/docker`
[2]`https://github.com/NERSC/shifter`

componente denominado *kubelet*, o qual também é utilizado para enviar dados relacionados ao monitoramento de contêineres ao nó principal.

E finalmente, o nó principal do Kubernetes mantém os componentes de gerenciamento, em que as informações acerca do *cluster* são persistidas num componente de armazenamento denominado *etcd* [Saito et al. 2016] – componente que implementa todo o armazenamento do Kubernetes. Neste nó, os componentes interagem uns com os outros por meio de APIs REST, onde os mesmos usam o servidor da API para salvar e recuperar informações. Por fim, a interação de um usuário com o ambiente de *cluster* é realizado pela interface de comando *kubectl*.

Conforme já fora mencionado, é importante salientar que, a despeito dos mecanismos providos pelo Kubernetes, o estado da aplicação hospedada no contêiner não é preservado/restaurado quando este é desativado. Neste sentido, este é o espaço de projeto que se busca explorar no âmbito deste trabalho.

3. Coordenação via Serviço no Kubernetes

Num ambiente de *cluster* onde se emprega a redundância para fins de melhoria da disponibilidade (i.e., tolerância a faltas) das aplicações, as requisições/pedidos que alteram o estado da(s) aplicação(ões) precisam ser devidamente coordenados antes de sua(s) respectiva(s) execução(ões). Uma abordagem bastante útil para realizar tal tarefa consiste na replicação de máquinas de estado (RME) [Schneider 1990]. Em suma, pelo uso da RME, pedidos concorrentes são submetidos a um algoritmo de consenso distribuído (p. ex.: Paxos [Lamport 1998] e Raft [Ongaro and Ousterhout 2014]) e a execução destes ocorre na mesma ordem em todas as réplicas.

É sabido que protocolos para RME disponíveis podem ser utilizados para realizar a coordenação dos pedidos de armazenamento. Entretanto, protocolos de coordenação, por mais simples ou eficientes que sejam, requerem esforço da aplicação para implementar sua integração. Cui e outros [Cui et al. 2015] descrevem a complexidade quanto ao uso de *interfaces* de aplicações como ZooKeeper [Hunt et al. 2010], para coordenar a replicação.

Uma possibilidade de utilização de um algoritmo de coordenação advém da incorporação desse algoritmo num ambiente existente. A literatura menciona que a incorporação de coordenação num ambiente pode ser feita pelo menos de três maneiras [Lung et al. 2000, Felber and Narasimhan 2004, Bessani et al. 2005]: integração, interceptação e serviço (também denominado *middleware*). No contexto de um ambiente, a abordagem de **integração** consiste em construir ou modificar um componente do ambiente, a fim de acrescentar uma funcionalidade.

De outro modo, a **interceptação** requer que as mensagens enviadas aos destinatários sejam capturadas e mapeadas num sistema de comunicação de grupo. É importante frisar que esse processo é feito de forma totalmente transparente à aplicação. Um exemplo prático consiste no uso de *interfaces* do sistema operacional [Narasimhan et al. 1997] para interceptar chamadas de sistema (i.e., *system calls*) relacionadas à aplicação.

E por fim, a abordagem de **serviço** consiste em definir uma camada de *software* entre o cliente e a aplicação, de modo que essa camada se torna responsável por prover a coordenação das requisições/pedidos enviada(o)s à aplicação.

3.1. Proposta de Arquitetura

Nesta seção se apresenta a proposta elaborada para realizar a coordenação da alteração de estados no Kubernetes, por meio da abordagem de serviço. Para tanto, foi desenvolvida uma arquitetura de sistema, doravante denominada CAIUS (**C**oorden**A**ção via Serv**I**ço no K**U**bernete**S**). Para facilitar a compreensão acerca da arquitetura proposta neste trabalho, a Figura 3(a) ilustra a execução coordenada de um pedido no âmbito do Kubernetes.

A operação normal do protocolo se dá da seguinte maneira. Inicialmente, o cliente envia o pedido (1), que por meio de um *firewall* (2) é entregue a um nó do *cluster* Kubernetes. O *proxy* do nó encaminha o pedido a uma réplica do contêiner de coordenação (3). O contêiner que recebeu o pedido realiza a ordenação do mesmo, na camada de coordenação via serviço (4). Após a ordem de execução ter sido estabelecida, o cliente é respondido com uma confirmação de que o pedido será em algum momento executado. Na sequência, por meio do *firewall* (5), a resposta é entregue ao cliente (6). E finalmente, o pedido entra em uma fila, e assim que for possível o mesmo é executado nas réplicas de aplicação (7).

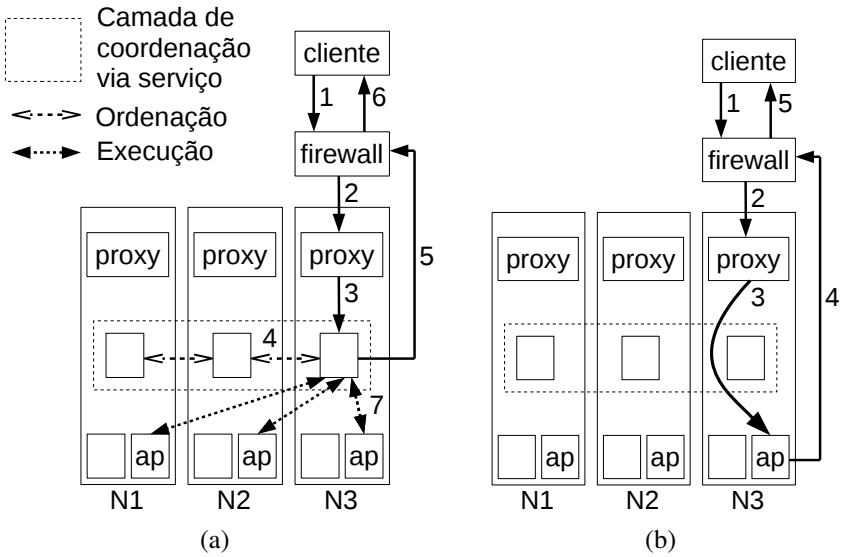


Figura 3. Kubernetes: incorporação de coordenação via serviço. (a) Atualizações coordenadas e (b) leituras diretamente na aplicação.

De outro modo, para os pedidos que efetuam apenas operações de leitura sobre estado da aplicação, é prevista uma otimização na execução do protocolo. Assim, os clientes que porventura vierem a executar operações de leitura sobre a aplicação, podem acessar diretamente as aplicações (Figura 3(b)), sem a necessidade de passar pelo serviço de coordenação. Neste caso, o cliente envia o pedido ao Kubernetes (1), que é entregue a um nó por meio do *firewall* (2). Na sequência, o *proxy* encaminha o pedido a uma das réplicas da aplicação (3), a aplicação executa o pedido (leitura de dados) e envia a resposta (4), que é entregue ao cliente (5) por meio do *firewall*.

Note pela Figura 3(a), que há uma separação das atividades/tarefas de ordenação e execução [Yin et al. 2003]. A separação destas tarefas em camadas distintas permite definir a consistência das operações, em função do número de réplicas da aplicação. Neste artigo, dois critérios de consistência são explorados para fins de análise, são eles: forte e eventual [Vogels 2009]. De maneira resumida, na consistência forte, depois que uma atualização de estado é concluída, qualquer acesso subsequente retornará o valor mais atual do estado. Um exemplo de aplicação que requer consistência forte é a edição colaborativa de documentos.

Sob outra perspectiva, a consistência eventual provê a garantia de que, após uma atualização do estado sem atualizações subsequentes, todos os leitores observarão o novo valor do estado. Assim, a janela de inconsistência pode ser determinada com base em fatores como latência de rede, carga de processamento e número de réplicas. Como exemplo, aplicações de rede social geralmente são satisfeitas com consistência eventual.

Na arquitetura proposta, pressupõe-se que a aplicação se encontra hospedada em contêineres. Diante disso, se houver apenas uma réplica da aplicação, toda operação de leitura retornará o último valor atualizado pelas operações de escrita coordenadas (Figura 3(b)) – esse cenário provê consistência forte à aplicação. Caso o número de réplicas da aplicação seja maior que um (1), que possivelmente é o caso, a fim de aumentar disponibilidade, o valor retornado pode ser o último valor completamente escrito (i.e., escrito em uma maioria de réplicas) ou um valor que esteja sendo escrito. Esse último cenário pode ocorrer em situações em que ocorram leituras e escritas simultâneas. Nesse caso, clientes da aplicação observam o estado sob uma consistência eventual.

3.2. Modelo de Sistema

Para a especificação da arquitetura, é pressuposto que o modelo de interação do ambiente é parcialmente síncrono [Dwork et al. 1988]. Para tanto, considera-se que clientes e réplicas são interconectados por uma rede em que os canais de comunicação são confiáveis (i.e., *reliable channels*), de modo que em algum momento, as mensagens são recebidas e entregues [Basu et al. 1996]. Em termos de faltas, são toleradas faltas por parada nas réplicas, em que réplicas podem parar de funcionar definitivamente, e assim deixar de realizar qualquer processamento ou comunicação.

Para fins de ordenação dos pedidos cuja finalidade é alterar o estado da aplicação, o CAIUS adota o Raft [Ongaro and Ousterhout 2014] como protocolo de consenso subjacente. Neste sentido, tão logo ocorre a ordenação de um pedido, o cliente é notificado pelo sistema por meio de um *ACK*. Os pedidos já ordenados entram numa fila de execução, cuja progressão se dá na medida em que uma maioria das réplicas da aplicação confirma a execução dos pedidos.

3.3. Controle de Fluxo de Execução

Para fins de compreensão, é importante salientar que o controle da execução dos pedidos nas réplicas de aplicação é feito pela réplica líder do Raft. No que segue, se apresenta uma explanação acerca do controle de fluxo realizado pela réplica líder. Note que a Figura 4 consiste num detalhamento da operação do líder, nos termos da Figura 3(a). Cada pedido recebido por um nó da camada de ordenação (passo 3) é ordenado pelo líder do protocolo subjacente de ordenação (Raft), em que este líder é que interage com as demais réplicas daquele protocolo (passo 4.1). Uma vez definida a ordem de um pedido, este é inserido numa fila de requisições (passo 4.2) e o cliente é notificado de que seu pedido será executado pelas réplicas da aplicação (passo 5). Para cada réplica de aplicação é criada uma *thread*, cuja finalidade destas é observar a fila de requisições (passo 7.1). Ao verificar um novo pedido na fila, a *thread* envia a respectiva requisição para a réplica de aplicação sob seu controle (passo 7.2). E finalmente, após a execução do pedido, a *thread* insere o resultado da execução no mapa de respostas (passo 7.3).

No que segue, são descritos alguns detalhes de operação do protocolo, aqui especificados pelos Algoritmos 1 e 2.

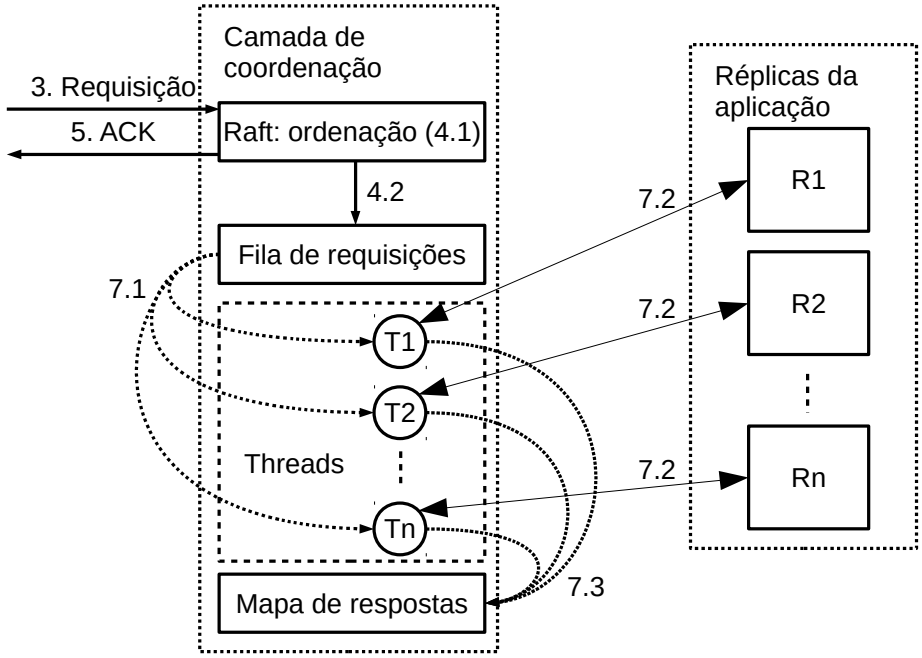


Figura 4. Estrutura interna do CAIUS

O serviço provido pela camada de coordenação é inicializado por meio da réplica líder do protocolo Raft, a partir da comunicação com as demais réplicas que fazem parte daquela instância do Raft (vide Algoritmo 1, linhas 1–10). A réplica líder do Raft contém uma fila para armazenar as requisições e um mapa para armazenar as respostas fornecidas às mesmas pelas réplicas da aplicação. Nesta réplica também são criadas as *threads* responsáveis por gerenciar a comunicação com cada nó da camada de aplicação (linhas 11-14 do Algoritmo 1). Os endereços das réplicas do Raft e da aplicação são obtidos pelas funções *onlineRaftReplicas* e *onlineAppReplicas*, respectivamente. Essas funções realizam chamadas à API do Kubernetes, em que é fornecida a *tag* dos contêineres para os quais deseja-se obter informações.

Algoritmo 1: Monitor de réplicas

```
1  answers[*,*] := {};
2  requestQueue := [];
3  lastAnswerdRequest := 0;
4  setOfRaftReplicas := {};
5  setOfApptReplicas := {};
6  while true do
7      for x | x in onlineRaftReplicas do
8          if x not in setOfRaftReplicas then
9              setOfRaftReplicas.append(x);
10             run raftReplica(x);
11     for x | x in onlineAppReplicas do
12         if x not in setOfAppReplicas then
13             setOfAppReplicas.append(x);
14             run appThread(x);
```

Note pelo Algoritmo 2, que durante a execução do protocolo as *threads* são mantidas num laço infinito, a fim de verificar a existência de novas requisições que chegam na

fila de requisições (Algoritmo 2, linha 4). Caso haja alguma nova requisição, esta é enviada pela *thread* para a respectiva réplica da aplicação sob o seu controle (linha 6), sendo que a resposta é adicionada no mapa que contém as respostas de cada requisição (linha 8). Assim, se a maioria das réplicas da aplicação responderam a requisição (linha 11), então essa requisição passa a demarcar, na fila de requisições, a última requisição respondida (linha 12).

Algoritmo 2: Controle de execução da aplicação

```
1  currentNode := x;
2  k := 0;
3  while true do
4      if len(requestQueue) > k then
5          k++;
6          request := requestQueue[k];
7          response := sendRequest(currentNode, request);
8          answers[k, currentNode] := response;
9          answerCount := len(answers[k]);
10         majority := floor(len(setOfReplicas) / 2) + 1;
11         if answerCount >= majority and lastAnswerdRequest < k then
12             lastAnswerdRequest := k;
```

4. Avaliação

No intuito de analisar o desempenho da arquitetura proposta, alguns experimentos foram realizados, cujos resultados são discutidos nesta seção. É importante salientar que a separação entre as camadas de ordenação e execução cria diferentes possibilidades de avaliação. Devido à semântica de execução das operações de atualização do estado no CAIUS (§ 3.2)[3] e da implementação adotada (§ 4.1), espera-se que o desempenho de acordo com o número de réplicas de ordenação ocorra segundo um padrão observado em trabalhos anteriores [Oliveira et al. 2016]. A execução de operações somente de escrita será realizada a fim de ratificar resultados já obtidos na literatura. Esse cenário de execução configura a especificação do primeiro experimento (e1).

Outra questão de interesse deste trabalho é aferir a diferença de desempenho ao variar o número de réplicas da aplicação, no contexto de operações de leitura do estado. O uso de uma ou mais réplicas de aplicação implica fornecimento de consistência forte ou eventual, para operações de leitura (§ 3). Um caso relevante, portanto, é o uso de apenas uma réplica de aplicação para verificar o comportamento da aplicação com consistência forte (e2). Outra possibilidade é o uso de três réplicas de aplicação, de forma a tolerar a falta, por parada, de uma das réplicas. O uso de um modelo de faltas por parada com ambiente assíncrono ($2f + 1$) permite maior disponibilidade (não impacta o desempenho durante situações de falta). Essa configuração (e3) provê à leitura do estado consistência eventual.

4.1. Ambiente e Implementação

O ambiente físico utilizado consistiu de um *cluster* composto por quatro microcomputadores, todos com as seguintes configurações: 1 (um) microprocessador Intel® Core™ i7

[3]O símbolo § significa referência a uma seção desse texto.

3.5GHz com 4 (quatro) núcleos e cache L3 8MB; 12GBytes de memória RAM, e; 1TB de armazenamento com 7200RPM. Para interconectar os elementos do *cluster* e cliente(s) foi utilizada uma rede de 10/100 Mbits, completamente isolada de tráfego externo. Como sistema operacional, adotou-se o Ubuntu 14.04.3 64 bits, *kernel* 3.19.0-42.

No ambiente do *cluster* utilizou-se o Kubernetes 1.1.7, de modo que uma máquina atuou como nó principal e as outras três máquinas atuaram como nós de execução de contêineres. Como implementação de contêineres foi utilizado o Docker. O cliente foi executado no nó principal. É digno de nota que o CAIUS foi implementado a partir da linguagem Go, versão 1.4.2. Como protocolo subjacente de ordenação, adotou-se o Raft. Nos experimentos, a aplicação avaliada consistiu num repositório de texto, denominada *logger*. A imagem dos contêineres do CAIUS e do *logger* estão disponíveis no Docker Hub[4], sob as *tags raft* e *logger*. Os respectivos códigos-fonte e os resultados dos experimentos (datados de 19/12/2016) estão disponíveis no GitHub[5].

No arquivo de configuração YAML dos contêineres do CAIUS para o Kubernetes, foi definida uma variável de ambiente que corresponde à *tag* dos contêineres da aplicação a ser coordenada. Nestes experimentos, seu valor foi especificado como "logger".

4.2. Experimentos

Em relação aos experimentos, a obtenção das amostras ocorreu a partir de três cenários de experimentação, os quais foram, nomeadamente: *i*) somente escritores, *ii*) leitores observando consistência forte, e, *iii*) leitores sujeitos a consistência eventual. Nos três experimentos houve atuação dos escritores. O primeiro experimento (e1) contou com três réplicas da aplicação. No segundo experimento (e2) foi criada apenas uma instância da aplicação, para garantir consistência forte aos clientes que efetivaram leitura diretamente da aplicação (sem passar pela coordenação). Por fim, no terceiro experimento (e3) foram criadas três réplicas da aplicação. Os clientes realizaram a leitura diretamente em apenas uma réplica da aplicação. Essa réplica foi escolhida pelo *proxy* do Kubernetes, a cada pedido, segundo o escalonamento padrão *Round-Robin*.

Não foi utilizado um *firewall* em nível de *cluster*. Todas as escritas e leituras foram direcionadas ao primeiro nó do *cluster*. No segundo experimento, a aplicação foi instanciada (pelo escalonador do Kubernetes) no primeiro nó. Em todos os experimentos o líder do Raft foi criado no segundo nó. A operação de escrita foi caracterizada por 8000 requisições de escrita, realizadas simultaneamente por 16 clientes. Operações de leitura (em e2 e e3) consistiram de 80.000 requisições de leitura, realizadas por 256 clientes simultâneos. As requisições de escrita e leitura foram executadas com uso da aplicação *ab*[6]. Tais requisições eram do tamanho de 100 bytes, enquanto que a(s) resposta(s) eram de 5 bytes. Os clientes enviavam novas requisições apenas após o recebimento da resposta da requisição anterior.

Durante os experimentos, os recursos consumidos foram monitorados com a ferramenta *dstat*[7]. Cada experimento foi monitorado pelo período de 35 segundos, tendo sido o monitoramento iniciado instantes antes do início da execução daquele experimento.

[4] `https://hub.docker.com/r/caiopo`

[5] `https://github.com/caiopo`, repositórios *raft* e *pontoon*.

[6] *Apache Benchmark*, em `http://httpd.apache.org/docs/2.4/programs/ab.html`.

[7] `dag.wiee.rs/home-made/dstat/`

4.3. Resultados e Discussões

A atuação de múltiplos clientes resultou em latências semelhantes percebidas por cada cliente escritor, em todos os experimentos (Tabela 1, segunda coluna). Esses valores estão nos limites conhecidos por experiências anteriores com o uso da implementação considerada neste artigo (42ms para 16 clientes, em [Oliveira et al. 2016]).

Tabela 1. Resultados dos experimentos.

	escritores (16)			leitores (256)		
	latência (ms)			latência (ms)		
experimento	média	desv.pad.	vazão (req/s)	média	desv.pad.	vazão (req/s)
e1	39.2	1.85	407	-	-	-
e2	42.3	28.54	373	4.95	19.17	50969
e3	42.58	21.86	375	7.64	29.68	33537

A latência percebida foi menor para clientes acessando a aplicação sob consistência forte. Apesar de os desvios-padrão serem altos, é digno que nota que um teste estatístico de hipótese (*t* de *Student*, com intervalo de confiança e 99%) mostra que a relação mencionada é verdadeira. Porém, essa vantagem ocorreu devido ao fato de que todas as leituras foram realizadas no mesmo nó onde a aplicação foi instanciada.

Clientes leitores que atuam sob a consistência eventual têm suas requisições sujeitas à ação do balanceamento de carga promovido pelo *proxy* em nível de nó. Devido a isso, percebem maior latência e menor vazão.

É importante salientar que os experimentos visam, sobretudo, demonstrar o consumo de recursos observado acerca da coordenação efetuada sobre os contêineres. Diante disso, no primeiro experimento (Figura 5), observa-se o alto consumo de recursos (rede e CPU) no segundo nó. Isso se deve ao fato do líder do Raft estar instanciado nesse nó. Além disso, corrobora o fato de o CAIUS também estar sendo executado junto ao líder do Raft.

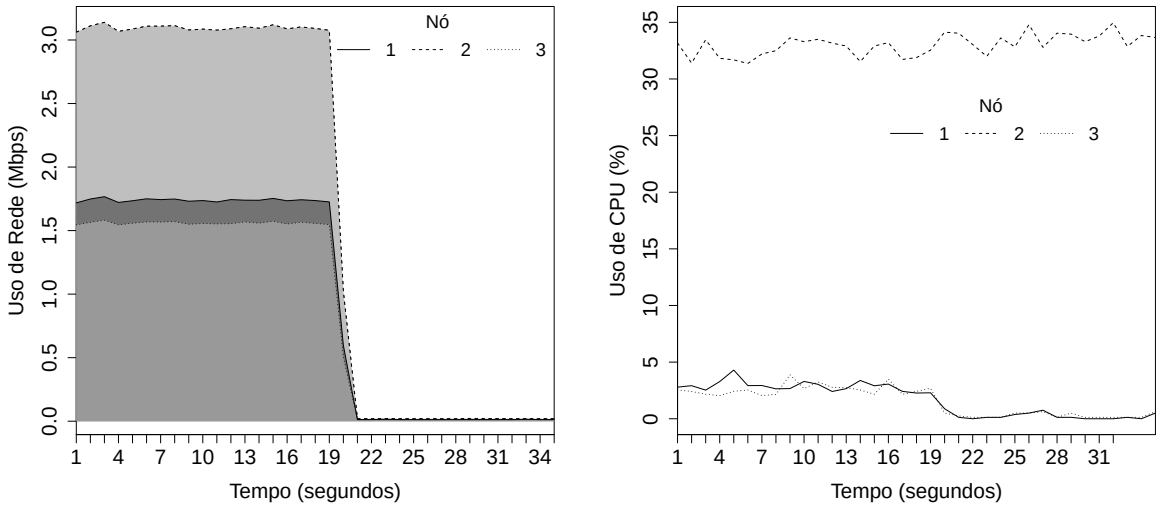
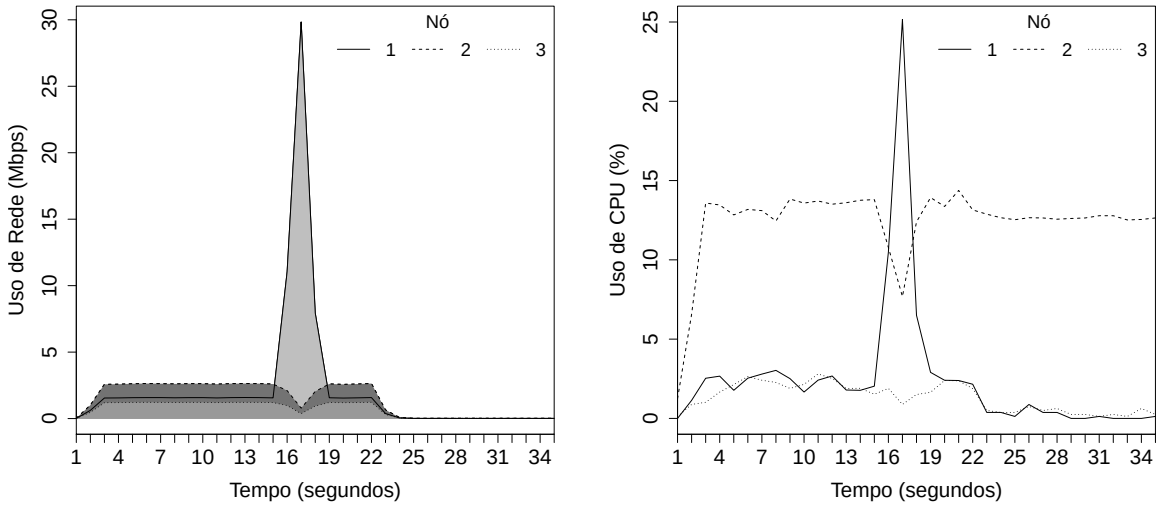


Figura 5. Experimento 1: consumo de (a) rede e (b) CPU.

De outro modo, no segundo experimento (Figura 6), o consumo de recursos é evidenciado na atuação dos clientes leitores. Entre os instantes 14s e 19s os clientes realizam a

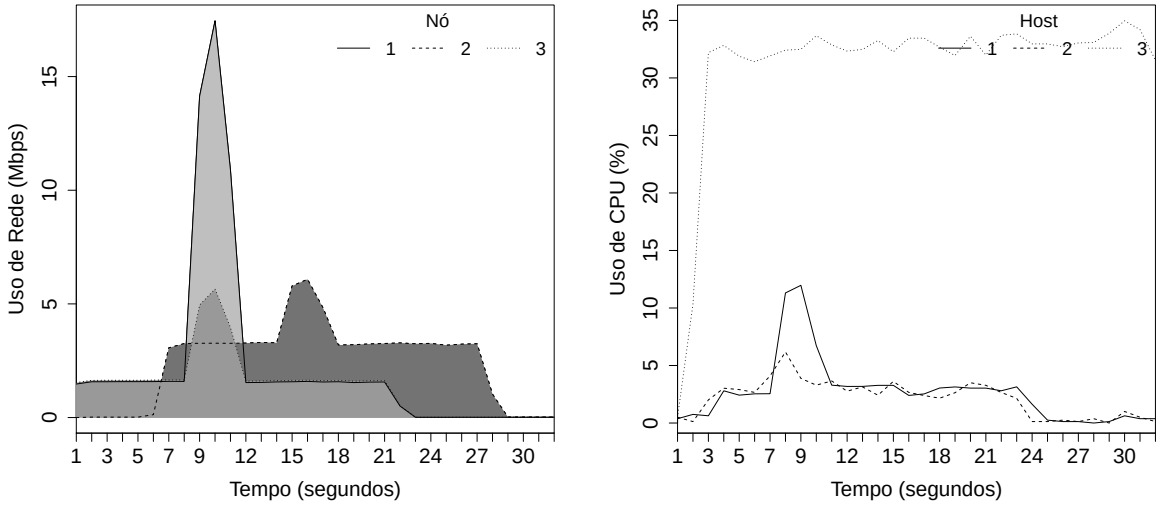
leitura de dados diretamente na réplica da aplicação. Esta é a razão pela qual ocorre o pico de consumo no primeiro nó.



(a) (b)

Figura 6. Experimento 2: consumo de (a) rede e (b) CPU.

E finalmente, o terceiro experimento (Figura 7) demonstra que o consumo de rede também manifesta um ponto de destaque no instante em que há a atuação de clientes leitores. Entretanto, ao que se percebe, o primeiro nó forneceu muito mais respostas do que os demais nós. Conforme mencionado sobre o aspecto de latência, o consumo predominou no primeiro nó devido ao fato de todos os pedidos de leitura terem sido realizados nesse nó.



(a) (b)

Figura 7. Experimento 3: consumo de (a) rede e (b) CPU.

5. Trabalhos Relacionados

A literatura dispõe de várias ferramentas que visam auxiliar na implementação de RME. Por exemplo, o BFT-SMaRt [Bessani et al. 2014] é disponível e implementa funcionalidades como transferência de estado e reconfiguração. O BFT-SMaRt já fora avaliado no cenário de contêineres [Torresini et al. 2016], onde aferiu-se novamente (corroborando [Felber and Narasimhan 2004]) a superioridade de desempenho dos contêineres em relação às tradicionais máquinas virtuais.

Em se tratando da atividade de coordenação, a literatura descreve que a incorporação de tal funcionalidade em ambientes pode ser feita por meio de integração, interceptação ou serviço [Lung et al. 2000]. O CRANE [Cui et al. 2015] torna transparente a coordenação realizada com o Paxos [Lamport 1998], por meio de interceptação, em nível de *sockets*, dos pedidos. No CAIUS a coordenação é acoplada ao Kubernetes por meio de um serviço, de modo que ela permanece transparente sob o ponto de vista da aplicação.

No contexto do Kubernetes, a RME já foi integrada ao mesmo, com vistas a prover a coordenação transparente para o usuário e reduzir o tamanho dos contêineres da aplicação [Netto et al. 2016]. Especificamente, a atuação do Raft no Kubernetes também já foi alvo de [Oliveira et al. 2016], onde se observou uma diferença de aproximadamente 17.4% de desempenho ao executar o Raft em ambiente virtualizado pelo Kubernetes (com uso do Docker) em comparação ao mesmo sendo executado diretamente numa máquina física. Porém, é digno de nota que a sobrecarga imposta pela virtualização é compensada pelas vantagens do gerenciamento que o Kubernetes proporciona ao ambiente.

6. Conclusões

A virtualização em nível de sistema (contêineres) é uma tendência nos provedores de nuvem. De maneira similar, ocorre o desenvolvimento de um ambiente para a gerência de contêineres (o Kubernetes) com vistas à adoção do mesmo por um grupo de provedores em nuvem. Dentre as aplicações que rodam em nuvem, algumas delas requerem sincronismo de dados, quando a aplicação mantém o estado da aplicação replicado. Este trabalho apresentou o CAIUS, uma arquitetura que provê coordenação a aplicações que necessitam ter o estado replicado. O CAIUS foi avaliado com aplicações que observaram as semânticas forte e eventual, duas semânticas bastante praticadas em aplicações da Internet.

O CAIUS é flexível ao ponto de poder ser modificado para utilizar outros algoritmos de consenso/ordenação, além do Raft. Por exemplo, o EPaxos [Moraru et al. 2013] não utiliza líder e pode ser uma alternativa que faz melhor uso do balanceamento provido pelo Kubernetes. Nesse caso, o CAIUS seria ativo em todas as réplicas da camada de coordenação, e não apenas no líder. Outra possibilidade de investigação futura é verificar os impactos da variação do número de réplicas que atuam na coordenação e o número de réplicas da aplicação. Adicionalmente, a realização de mais experimentos considerando o balanceamento de carga em nível de *cluster* (provido pelo *firewall*) torna-se fundamental para instanciar um cenário mais próximo da realidade em provedores, nos quais as escritas e leituras de dados podem ser realizadas em quaisquer réplicas da camada de coordenação e de aplicação, respectivamente.

Agradecimentos

Este trabalho foi parcialmente financiado pela FAPESC/IFC, projeto nº 00001905/2015. Infelizmente, durante a elaboração deste trabalho o quarto autor, nosso orientador, colega e amigo Lau Cheuk Lung faleceu devido a súbitas complicações de saúde. Aproveitamos essa oportunidade para prestar uma pequena homenagem a um grande pesquisador e professor, que influenciou muitos alunos e colegas durante sua carreira.

Referências

Basu, A., Charron-Bost, B., and Toueg, S. (1996). Simulating reliable links with unreliable links in the presence of process crashes. In *Proceedings of the 10th International Workshop on Distributed Algorithms*, pages 105–122.

Bernstein, D. (2014). Containers and cloud: From LXC to docker to Kubernetes. *IEEE Cloud Computing*, 1(3):81–84.

Bessani, A., Sousa, J., and Alchieri, E. E. (2014). State machine replication for the masses with bft-smart. In *Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 355–362.

Bessani, A. N., Lung, L. C., and da Silva Fraga, J. (2005). Extending the UMIOP specification for reliable multicast in CORBA. In *Proceedings of the nternational Symposium on Distributed Objects and Applications*, pages 662–679.

Cui, H., Gu, R., Liu, C., Chen, T., and Yang, J. (2015). Paxos made transparent. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 105–120. ACM.

Dwork, C., Lynch, N. A., and Stockmeyer, L. (1988). Consensus in the presence of partial synchrony. *Journal of ACM*, 35(2):288–322.

Felber, P. and Narasimhan, P. (2004). Experiences, strategies, and challenges in building fault-tolerant CORBA systems. *Transactions on Computers*, 53(5):497–511.

Hunt, P., Konar, M., Junqueira, F. P., and Reed, B. (2010). Zookeeper: Wait-free coordination for internet-scale systems. In *Proceedings of the USENIX Annual Technical Conference*, page 9.

Lamport, L. (1998). The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169.

Lung, L. C., da Silva Fraga, J., Farines, J.-M., and de Oliveira, J. R. S. (2000). Experiências com comunicação de grupo nas especificações fault tolerant CORBA. In *Simpósio Brasileiro de Redes de Computadores*.

Mell, P. and Grance, T. (2011). The NIST definition of cloud computing. Technical report, National Institute of Standards and Technology Gaithersburg.

Moraru, I., Andersen, D. G., and Kaminsky, M. (2013). There is more consensus in egalitarian parliaments. In *Proceedings of the Twenty-Fourth Symposium on Operating Systems Principles*, pages 358–372. ACM.

Narasimhan, P., Moser, L. E., and Melliar-Smith, P. M. (1997). Exploiting the internet inter-ORB protocol interface to provide CORBA with fault tolerance. In *Proceedings of the 3rd USENIX Conference on Object-Oriented Technologies and Systems*, pages 6–15.

Netto, H. V., Lung, L. C., Correia, M., and Luiz, A. F. (2016). Replicação de máquinas de estado em *containers* no kubernetes: uma proposta de integração. In *Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuidos*.

Oliveira, C., Lung, L. C., Netto, H., and Rech, L. (2016). Evaluating raft in docker on kubernetes. In *Proceedings of the International Conference on Systems Science*, pages 123–130.

Ongaro, D. and Ousterhout, J. (2014). In search of an understandable consensus algorithm. In *Proceedings of the USENIX Annual Technical Conference*, pages 305–319.

Parmelee, R. P., Peterson, T. I., Tillman, C. C., and Hatfield, D. J. (1972). Virtual storage and virtual machine concepts. *IBM Systems Journal*, 11(2):99–130.

Popek, G. J. and Goldberg, R. P. (1974). Formal requirements for virtualizable third generation architectures. *Communications of the ACM*, 17(7):412–421.

Saito, H., Lee, H.-C. C., and Hsu, K.-J. C. (2016). *Kubernetes Cookbook*. Packt Publishing, Aachen, DE.

Schneider, F. B. (1990). Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319.

Smith, J. E. and Nair, R. (2005). The architecture of virtual machines. *Computer*, 38(5):32–38.

Soltesz, S., Pötzl, H., Fiuczynski, M. E., Bavier, A., and Peterson, L. (2007). Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. In *Proceedings of the 2nd ACM SIGOPS European Conference on Computer Systems*, pages 275–287.

Torresini, E., Pacheco, L. A., Alchieri, E. A. P., and Caetano, M. F. (2016). Aspectos práticos da virtualização de replicação de máquina de estado. In *Workshop on Cloud Networks & Cloudscape Brazil, Congresso da Sociedade Brasileira de Computação*.

Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., and Wilkes, J. (2015). Large-scale cluster management at Google with Borg. In *Proceedings of the 10th European Conference on Computer Systems*, pages 18:1–18:17.

Vogels, W. (2009). Eventually consistent. *Communications of the ACM*, 52(1):40–44.

Yin, J., Martin, J.-P., Venkataramani, A., Alvisi, L., and Dahlin, M. (2003). Separating agreement from execution for Byzantine fault tolerant services. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pages 253–267.

Um Mecanismo para Compartilhamento de Recursos em Nuvens Colaborativas Baseado na Credibilidade dos Usuários

Hugo Sadok, Miguel Elias M. Campista e Luís Henrique M. K. Costa *

[1]Universidade Federal do Rio de Janeiro - PEE/COPPE/GTA - DEL/POLI

{sadok, miguel, luish}@gta.ufrj.br

Resumo. *As nuvens colaborativas permitem que usuários com poucos recursos obtenham as mesmas vantagens de elasticidade das nuvens convencionais, ao contribuir com seus recursos ociosos. No entanto, quando a demanda por recursos é maior que a oferta, conflitos surgem e é necessário definir a alocação de cada usuário. Este trabalho propõe um mecanismo de alocação que garante a eficiência de utilização dos recursos ao mesmo tempo em que incentiva a colaboração. As propriedades do mecanismo proposto são demonstradas teoricamente, utilizando teoria dos jogos, a partir da definição de um modelo baseado em um cenário Bayesiano. Neste cenário os usuários constantemente decidem a quantidade de recursos que irão fornecer, ou requisitar, da nuvem colaborativa. Além disso, o mecanismo é verificado com simulações conduzidas através de traços reais da utilização da nuvem do Google. Os resultados mostram um aumento no número de requisições atendidas de até aproximadamente três vezes em comparação com a não utilização do mecanismo.*

Abstract. *Collaborative clouds bring the elasticity advantages from traditional clouds to users with few resources, as long as they contribute with their idle resources. Nevertheless, when demand for resources exceeds supply, conflicts arise and each user allocation must be defined. In this work, we propose an allocation mechanism that guarantees resource utilization efficiency while encouraging collaboration. Using game theory, we demonstrate the mechanism properties by defining a model based on a Bayesian setting. In such setting, users constantly decide the amount of resources to either contribute, or request, from the collaborative cloud. Moreover, we use simulations with the Google cloud dataset to verify the mechanism. The results show an increase in the number of served requests up to approximately three times compared to not using the mechanism.*

1. Introdução

O paradigma de computação em nuvem se baseia no provisionamento de recursos computacionais em centros de dados acessíveis pela rede. Além das vantagens trazidas pela terceirização da operação da infraestrutura, a nuvem fornece aos usuários maior flexibilização do dimensionamento da infraestrutura necessária às suas aplicações [Costa et al., 2012]. Sem a nuvem, os usuários precisam dimensionar seus recursos para atender os possíveis picos de demanda, causando ociosidade na maior parte

*Este trabalho foi parcialmente financiado pela CAPES, CNPq, FAPERJ e FINEP.

do tempo. Além disso, os provedores de nuvem conseguem oferecer serviços a preços competitivos por se beneficiarem da agregação e heterogeneidade de uso de vários usuários [Armbrust et al., 2010]. Entretanto, mesmo com o cenário de crescente popularização da computação em nuvem, muitos usuários (empresas, laboratórios, etc.) possuem sua própria infraestrutura computacional, seja por possuir uma base já instalada, seja por uma política de aquisição de equipamentos da instituição. A computação em nuvem neste caso se torna uma forma desses usuários obterem mais recursos quando necessário, e ao mesmo tempo compartilharem recursos de forma eficiente.

Algumas arquiteturas em nuvem são utilizadas para prover elasticidade, dentre elas a nuvem híbrida [Sotomayor et al., 2009]. Nuvens híbridas integram a infraestrutura local a uma nuvem pública, de forma que os usuários possam obter recursos adicionais em casos de pico de demanda. Uma pesquisa feita em 2016 com 1060 profissionais de TI mostrou que 71% usavam uma nuvem híbrida [Weins, 2016]. Embora os usuários possam obter recursos extras em caso de pico, é comum a presença de recursos ociosos. Uma alternativa à nuvem híbrida é o paradigma de nuvem colaborativa. Uma nuvem colaborativa é formada por máquinas dedicadas, cedidas pelos próprios usuários da nuvem. Esses usuários frequentemente estão geograficamente separados, dando um aspecto distribuído à nuvem. Os integrantes possuem acesso aos recursos ociosos dos demais, em seus momentos de pico, e contribuem com seus próprios recursos, quando ociosos. A nuvem colaborativa traz os benefícios da agregação e da heterogeneidade de uso, características das nuvens tradicionais, ao mesmo tempo em que permite que os usuários usem sua própria infraestrutura, como nas nuvens híbridas. Diferente do conceito de grades de computação, em que os usuários combinam recursos com um objetivo comum [Vaquero et al., 2008], os usuários da nuvem colaborativa podem ter objetivos distintos. Um exemplo de nuvem colaborativa é o Projeto GT-PID [Couto et al., 2015].

Embora as vantagens da agregação de recursos na nuvem colaborativa sejam facilmente vislumbradas, as decisões de alocação de recursos entre os usuários não são. No cenário onde os recursos de todos os usuários estão agregados em uma nuvem colaborativa, surgem as questões: Em caso de sobre-demanda, quais usuários devem receber mais recursos? Como garantir o interesse dos usuários em disponibilizar seus recursos ociosos? Este trabalho propõe uma solução baseada em princípios da teoria dos jogos. A solução envolve um modelo onde, a cada iteração, os usuários decidem a quantidade de recursos que desejam contribuir ou receber. Com base neste modelo, foi criado um mecanismo para gerir as alocações. Quando há mais recursos sendo ofertados do que pedidos, todos os pedidos são atendidos. Já quando os pedidos superam os recursos ofertados, as alocações são feitas baseadas nas credibilidades dos usuários, as quais são calculadas a partir das contribuições passadas. Este trabalho apresenta as seguintes contribuições, confirmadas a partir de demonstrações teóricas e de traços reais da nuvem do Google:

- **Um modelo de satisfação dos usuários de uma nuvem colaborativa.** O modelo define a satisfação em relação à quantidade de recursos recebidos de acordo com as necessidades dos usuários. O modelo também simplifica o projeto de mecanismos ao abstrair os recursos do usuário na nuvem e focar somente na quantidade de recursos ociosos e nos desejos dos usuários.
- **Um mecanismo de colaboração de recursos entre os usuários da nuvem.** O mecanismo garante que os usuários com maior contribuição no passado obtêm

mais recursos em caso de conflito, o que também cria um incentivo à colaboração. Mesmo com o uso da contribuição passada, o mecanismo possibilita a entrada de novos usuários. Além disso o mecanismo incentiva a verdade, de modo que os usuários não obtêm satisfação melhor ao mentir sobre suas necessidades.
- **Um algoritmo que rege as decisões de alocação do mecanismo.** O algoritmo se baseia nas restrições impostas pelo mecanismo para realizar a troca de recursos com base nas credibilidades dos usuários.

Este artigo está organizado da seguinte forma. A Seção 2 mostra uma visão geral do mecanismo e seus requisitos. A Seção 3 introduz o modelo de compartilhamento de recursos entre os usuários da nuvem. A Seção 4 propõe o mecanismo de alocações o qual é avaliado nas Seções 5 e 6. A Seção 7 descreve os trabalhos relacionados. Finalmente, a Seção 8 conclui o trabalho e aponta direções futuras.

2. Visão Geral do Mecanismo de Alocação

Nas nuvens tradicionais, os provedores se beneficiam da chamada multiplexação estatística, em que a necessidade por recursos agregada de diversos usuários evita que o provedor tenha que dimensionar a nuvem para atender todos os picos ao mesmo tempo [Armbrust et al., 2010]. Na nuvem colaborativa também ocorre a multiplexação estatística, mas isso evita que os próprios usuários precisem dimensionar seus recursos para o pico. Contudo, diferente das nuvens tradicionais, o caráter gratuito facilita que os recursos da nuvem colaborativa se esgotem. É essencial a presença de um mecanismo que decida as alocações quando a demanda por recursos supera a oferta. O mecanismo proposto neste trabalho evita o uso desmedido de recursos ao considerar os comportamentos passados no cálculo da credibilidade de cada usuário. Usuários que requisitam mais recursos do que fornecem são gradualmente penalizados nas decisões de alocação. Por outro lado, usuários com maiores contribuições no passado têm vantagens no conflito por recursos. O mecanismo funciona de acordo com os seguintes requisitos:

- **Racionalidade Individual**: As decisões de alocação devem ser cuidadosas de modo que nenhum usuário se prejudique em usar a nuvem colaborativa. Caso um usuário tivesse acesso a menos recursos do que teria sem fazer parte da nuvem, a decisão mais racional seria abandonar o sistema.
- **Eficiência Estrita de Pareto**: Os recursos do sistema devem ser usados de forma eficiente. O mecanismo deve garantir que o máximo de recursos do sistema seja utilizado.
- **Verdade**: Os usuários da nuvem não devem conseguir manipular o mecanismo de alocação ao mentir sobre suas necessidades por recursos.
- **Incentivo à Colaboração**: Os usuários devem obter benefícios por compartilhar seus recursos ociosos.
- **Competitividade de Novos Entrantes**: Usuários novos devem ter condição de competir por recursos do sistema, caso contrário, eles não se juntariam à nuvem.

O mecanismo proposto funciona em iterações. A cada iteração, os usuários expressam seu desejo de contribuir ou requisitar recursos. De acordo com os desejos e credibilidades de todos os usuários, o mecanismo pode decidir uma alocação de recursos. A credibilidade de cada usuário representa sua tendência em contribuir ou requisitar recursos. Considere que um usuário com credibilidade "-2" costuma contribuir 2 recursos, e

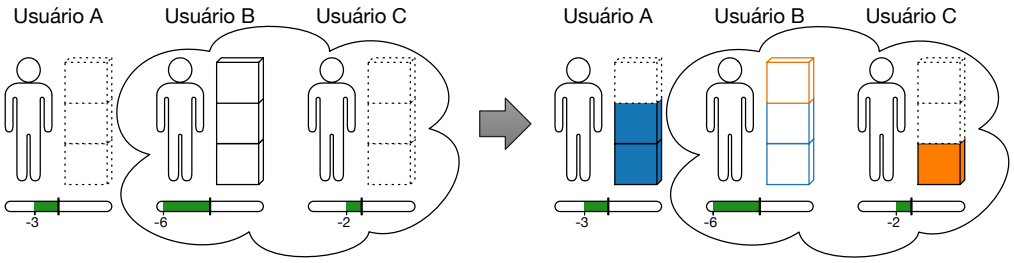


Figura 1. Exemplo da visão que o Usuário A tem da nuvem colaborativa. Os Usuários A e C precisam de 3 recursos cada, enquanto o Usuário B oferece 3 recursos. Como a credibilidade de A é -3 (barra verde inferior) e a de C é -2, o Usuário A recebe 2 recursos enquanto C recebe 1 recurso (1 a menos que A).

um usuário com credibilidade "-3" costuma contribuir 3 recursos. É natural assumir que, quando há sobrecarga da nuvem, o usuário que costuma contribuir com 3 recursos deve receber preferencialmente 1 recurso a mais em relação ao que costuma contribuir com 2 recursos. Essa intuição é a base do algoritmo de alocação proposto e está exemplificada na Figura 1.

3. Modelo de Compartilhamento de Recursos

No modelo, a utilização do sistema é dividida em intervalos de tempo de mesma duração. Em cada intervalo ocorre uma iteração em que os usuários escolhem a quantidade de recursos que desejam fornecer ou requisitar do sistema. O conjunto de usuários é denotado por $\mathcal{N}$, tal que $|\mathcal{N}| = N$. As escolhas dos usuários são representadas pelas chamadas ações. Uma ação de um usuário $i \in \mathcal{N}$ é denotada por a_i. Ações positivas representam pedidos por recursos enquanto que ações negativas representam contribuições. Como é impossível que o sistema saiba se a ação do usuário representa de fato o seu desejo (um usuário pode precisar de 2 recursos mas pedir 4 para o sistema), cada usuário possui um *tipo* que representa seu real desejo em contribuir ou requisitar recursos. O tipo de um usuário i é denotado por θ_i e só é conhecido por este usuário. Observe que se o usuário i revela a verdade, sua ação é igual ao seu tipo ($a_i = \theta_i$).

O modelo também assume a presença de um mecanismo. Com base nas ações dos usuários, o mecanismo deve ser capaz de decidir uma alocação de recursos. Um usuário i, recebe uma alocação de recursos o_i. Se o mecanismo decidiu que o usuário receberá recursos do sistema, sua alocação é positiva. Por outro lado, se o usuário contribuirá recursos para o sistema, sua alocação é negativa. Dependendo da alocação do mecanismo os usuários podem ficar satisfeitos ou não. Para cada usuário é definida também uma função utilidade. Denota-se u_i a função utilidade do usuário i. Quando a função utilidade retorna um valor positivo, o usuário está mais satisfeito do que estaria sem fazer parte da nuvem. Alternativamente, caso a função retorne um valor negativo, o usuário estaria mais satisfeito sem usar a nuvem. A seguir, as variáveis do modelo são definidas formalmente. O resumo das notações pode ser visto na Tabela 1.

3.1. Formalização do Modelo

O modelo é definido como um jogo Bayesiano, sendo dividido em cenário e mecanismo. Tudo que é definido no cenário é fixo e não pode ser redefinido pelo mecanismo. O cenário é modelado como uma tupla $(\mathcal{N}, \mathcal{O}, \Theta, p, u)$. $\mathcal{N}$ foi definido anteriormente e

Tabela 1. Resumo das notações usadas no modelo.

Notação	Descrição
N	Número de usuários
$\mathcal{N}$	Conjunto de todos os usuários
$\mathcal{O}$	Conjunto de alocações possíveis
o	Uma alocação de recursos em $\mathcal{O}$ para todos os usuários, $o = (o_1, \ldots, o_N)$
o_i	Quantidade de recursos alocados para i
Θ	Combinação de tipos possíveis de todos os usuários, $\Theta = \Theta_1 \times \ldots \times \Theta_N$
Θ_i	Conjunto de tipos possíveis para o usuário i
θ	Escolha de tipos de todos os usuários, $\theta \in \Theta$
θ_i	Um tipo de i, $\theta_i \in \Theta_i$
r_i	Quantidade de recursos que i possui
p	Distribuição de probabilidade dos tipos em Θ
u	Funções utilidade dos usuários, $u = (u_1, \ldots, u_N)$
$u_i(o, \theta_i)$	Função utilidade do usuário i
$\mathcal{A}$	Combinação de ações de todos os usuários, $\mathcal{A} = A_1 \times \ldots \times A_N$
A_i	Conjunto de ações possíveis para o usuário i
a	Ações escolhidas por todos os usuários, $a \in \mathcal{A}$
a_i	Ação escolhida por i
$\mathcal{M}(a)$	Função Mecanismo

representa o conjunto de usuários. $\mathcal{O}$ é o conjunto de alocações possíveis de forma que $(o_1, \ldots, o_N) \in \mathcal{O}$. As alocações de recursos devem garantir que a quantidade de recursos recebidos pelos usuários não ultrapasse a quantidade de recursos fornecidos. Além disso, é impossível que um usuário i forneça mais recursos do que de fato possui. Considerando que r_i representa a quantidade de recursos que o usuário i possui, o conjunto $\mathcal{O}$ pode ser definido como:

$$\mathcal{O} = \left\{ (o_1, \ldots, o_N) \in \mathbb{Z}^N \middle| (\forall i \in \mathcal{N}, o_i \geq -r_i) \wedge \sum_{i \in \mathcal{N}} o_i \leq 0 \right\}. \quad (1)$$

Θ é definida como $\Theta = \bigtimes_{i=1}^{N} \Theta_i$ em que Θ_i são os tipos possíveis para o usuário i. Os tipos dos usuários são limitados inferiormente pela quantidade de recursos que estes possuem de forma que $\Theta_i = \{\theta_i \in \mathbb{Z} | \theta_i \geq -r_i\}$. p representa a distribuição de probabilidade dos tipos em Θ. Finalmente, u representa uma tupla com as funções utilidade de todos os usuários, $u = (u_1, \ldots, u_N)$. A função utilidade do usuário i é definida como:

$$u_i(o_i, \theta_i) = \begin{cases} \min(o_i, \theta_i), \text{ para } \theta_i \geq 0. \\ \min(0, o_i - \theta_i), \text{ para } \theta_i < 0. \end{cases} \quad (2)$$

Observe na Equação 2 que quando um usuário precisa de mais recursos ($\theta_i > 0$) ele tem utilidade negativa caso $o_i < 0$ e utilidade positiva caso $o_i > 0$. No entanto a utilidade é limitada superiormente por θ_i, de modo que o usuário não se beneficia de mais recursos do que precisa. Alternativamente, quando um usuário possui recursos ociosos ($\theta_i < 0$) ele não pode obter utilidade positiva, mas obtem utilidade negativa caso $o_i < \theta_i$, ou seja, caso contribua com mais recursos do que gostaria. No caso intermediário em que $\theta_i = 0$, os usuários são indiferentes em receber recursos mas têm utilidade negativa caso contribuam com recursos para o sistema.

O mecanismo é um par $(\mathcal{A}, \mathcal{M})$. O conjunto $\mathcal{A}$ representa as possíveis escolhas de ações dos usuários do mecanismo. Como dito anteriormente, as ações dos usuários podem ou não ser iguais aos seus tipos. Ainda assim, as ações possíveis devem ser as mesmas que os tipos possíveis de forma que $\mathcal{A} = \Theta$.[1] $\mathcal{M}$ é a função mecanismo que mapeia as ações dos usuários em alocações de recursos ($\mathcal{M} : \mathcal{A} \rightarrow \mathcal{O}$). A função mecanismo independe da definição do modelo e será definida a seguir.

4. Mecanismo de Alocação

O mecanismo decide uma alocação de recursos válida a partir dos desejos dos usuários de receber ou contribuir. Para que haja um incentivo à contribuição de recursos ociosos, o mecanismo deve garantir que essa alocação privilegie usuários com maior contribuição no passado. As contribuições passadas são representadas pela credibilidade de cada usuário. No entanto, deve-se projetar o mecanismo de forma que também possibilite a entrada de novos usuários.

4.1. Credibilidade dos Usuários

A credibilidade do usuário i no instante t, representada por $c_i(t)$, é calculada usando uma média móvel exponencial das alocações passadas, definida de forma recursiva como: $c_i(t) = (1 - \delta) \cdot o_i(t) + \delta \cdot c_i(t - 1)$. A credibilidade usa o parâmetro $\delta \in [0, 1)$, o qual define o quanto as amostras passadas influenciam na credibilidade. Para $\delta = 0$, a credibilidade só considera a última alocação de recursos ($o_i(t)$). Quanto mais δ se aproxima de 1, mais os valores de alocações passadas influenciam na credibilidade. O amortecimento da credibilidade garante que novos usuários da nuvem, os quais entram no sistema com credibilidade nula, tenham condição de disputar recursos com os usuários mais antigos. Existe, contudo, um compromisso na escolha do valor de δ. Valores muito pequenos fazem com que o sistema não recompense suficientemente os usuários por suas alocações passadas. Por outro lado, valores muito próximos de 1 fazem com que a resposta do sistema às mudanças de comportamento dos usuários seja lenta.

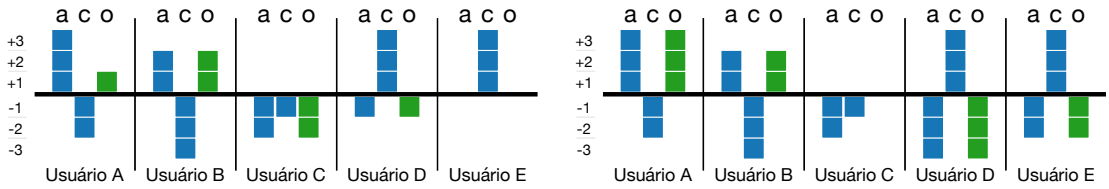
4.2. Restrição à Saída do Mecanismo

Embora o modelo limite as possíveis saídas do mecanismo ao conjunto $\mathcal{O}$, esse conjunto ainda admite a presença de algumas saídas indesejáveis de acordo com os requisitos do mecanismo. É conveniente definir um subconjunto $\Psi \subseteq \mathcal{O}$ com somente as alocações desejáveis:

$$\Psi = \left\{ (o_1, \ldots, o_N) \in \mathcal{O} \middle| \sum_{i \in \mathcal{N}} o_i = 0 \wedge \forall i \in \mathcal{N}, \min(a_i, 0) \leq o_i \leq \max(a_i, 0) \right\}. \quad (3)$$

A primeira restrição é contra o desperdício de recursos. Não é permitido que mais recursos sejam contribuídos do que utilizados. Ou seja, a alocação deve obedecer $\sum_{i \in \mathcal{N}} o_i = 0$. Outra restrição que se impõe tem a finalidade de garantir a racionalidade individual. Para isso, basta impedir alocações que gerem utilidade negativa, para qualquer usuário. Isso é feito ao garantir que $o_i \geq \min(\theta_i, 0), \forall i$. Para $a_i = \theta_i$, isso equivale a dizer

[1] Mecanismos onde $\mathcal{A} = \Theta$ são chamados em teoria dos jogos de mecanismos diretos. O princípio da revelação garante que não há perda de generalidade ao se impor essa restrição [Myerson, 1982].



(a) Demanda maior que oferta.

(b) Oferta maior que demanda.

Figura 2. Alocações (o) geradas pelo mecanismo a partir das ações (a) e credibilidades (c) dos usuários A até E. Ações positivas representam demanda, enquanto negativas representam oferta. Credibilidades negativas indicam que, no passado, o usuário contribuiu mais recursos do que requisitou.

que $o_i \geq \min(a_i, 0)$. Uma outra forma de desperdício de recursos é atribuir mais recursos a um usuário do que ele de fato precisa. Para impedir que isso ocorra, também com $a_i = \theta_i$, basta garantir que $o_i \leq \max(a_i, 0)$. A função mecanismo será então definida de forma a gerar saídas no conjunto Ψ.

4.3. Função Mecanismo

A intuição para a definição da função mecanismo está em inferir, através da credibilidade, a quantidade de recursos que um usuário costuma contribuir ou requisitar do sistema. Os usuários que costumam contribuir mais devem receber mais. Por outro lado, usuários que costumam contribuir menos poderão compensar futuramente com maior contribuição.

Ao garantir vantagens para os usuários que contribuíram mais no passado, surge um outro tipo de conflito: quando a oferta supera a demanda, quais usuários devem ser escolhidos para contribuir para o sistema e com isso ter uma melhora na credibilidade? Os dois tipos de conflito (oferta maior que demanda e demanda maior que oferta) são tratados pelo mecanismo de maneira simétrica. No caso do conflito pelo uso de recursos, o mecanismo privilegia os usuários com melhor credibilidade. No conflito pela oferta de recursos, o mecanismo privilegia os usuários com pior credibilidade. Assim, usuários que contribuíram mais no passado são preferencialmente recompensados com recursos. Já os que contribuíram com menos recursos têm maior chance de contribuir, podendo então melhorar suas credibilidades.

A Figura 2 ilustra um exemplo de dois cenários de conflito com cinco usuários de A até E. Para cada usuário, especifica-se os valores da ação (**a**), credibilidade (**c**) e alocação (**o**). No cenário onde a demanda por recursos é maior que a oferta (Figura 2(a)), existe demanda por 5 recursos, mas somente 3 recursos são oferecidos. O mecanismo cuida do conflito entre os usuários que precisam de recursos. Como o usuário B tem credibilidade melhor (mais negativa), ele contribuiu mais no passado e, portanto, recebe uma alocação de recursos maior. O mesmo exemplo pode ser adaptado de forma que a oferta supere a demanda (Figura 2(b)). Observe que agora todas as demandas são aceitas e o usuário C deixará de contribuir com recursos, já que sua credibilidade é melhor que a de D e de E, os quais conseguem suprir a demanda sozinhos.

Essas alocações podem ser computadas matematicamente como será mostrado a seguir. Considerando o primeiro exemplo (Figura 2(a)), há um vetor de ações $\boldsymbol{a} = [3, 2, -2, -1, 0]$ e um vetor de credibilidades, calculado a partir de alocações passadas

hipotéticas, $\boldsymbol{c} = [-2, -3, -1, 3, 3]$. O mecanismo usa $\boldsymbol{a}$ para definir o conjunto de alocações possíveis Ψ, conforme a Equação 3. Uma alocação $\boldsymbol{o} \in \Psi$ é então calculada de forma a minimizar a variância do vetor $\boldsymbol{c} + \boldsymbol{o}$ desde que garantindo a alocação da quantidade máxima de recursos. Em resumo, a função mecanismo resolve:

$$\underset{\boldsymbol{o} \in \Phi \subseteq \Psi}{\text{minimizar}} \sum_{i \in \mathcal{N}} (c_i + o_i)^2 \ \textit{(minimizar a variância)}, \tag{4}$$

tal que:

$$\Phi = \underset{\boldsymbol{o} \in \Psi}{\arg\max} \sum_{i \in \mathcal{N}} |o_i| \ \textit{(alocar a quantidade máxima de recursos)}. \tag{5}$$

O cálculo da variância não inclui a média do vetor $\boldsymbol{c} + \boldsymbol{o}$, já que a média é sempre nula[2]. Vale notar que minimizar a variância de $\boldsymbol{c} + \boldsymbol{o}$ neste caso equivale a minimizar os máximos de $\boldsymbol{c} + \boldsymbol{o}$ lexicograficamente (minimizar o primeiro máximo, o segundo e assim sucessivamente), quando a oferta é maior que a demanda, e maximizar os mínimos lexicograficamente, caso contrário. Para encontrar uma solução para a Equação 4 pode-se recorrer a métodos de otimização com restrições. No entanto, também é possível definir um algoritmo determinístico para resolver o mesmo problema.

4.4. Algoritmo de Alocação

As restrições criadas para a saída do mecanismo (Equações 3 e 5) fazem com que, quando a demanda é maior que a oferta, todas as ações de oferta são satisfeitas (se $a_i \leq 0$ então $o_i = a_i$). Já quando a oferta é maior que a demanda, todas as ações de demanda são satisfeitas (se $a_i \geq 0$ então $o_i = a_i$). Além de conveniente para o algoritmo, essa característica reforça a simetria entre os dois casos, sendo possível focar em apenas um deles. Quando a demanda é maior que a oferta $\left(\sum_{i \in \mathcal{N}} a_i > 0\right)$, o algoritmo precisa definir as alocações da parte em conflito ($a_i > 0$). A quantidade de recursos disponíveis a ser alocada para a parte em conflito é dada pela soma das ações negativas $\left(-\sum_{i \in \mathcal{N}} \min(a_i, 0)\right)$.

O Algoritmo 1 usa a ideia de maximizar os mínimos lexicograficamente. Para isso, faz-se uso de uma fila de prioridade Q, responsável por guardar as credibilidades dos usuários em conflito. Como as menores credibilidades são as melhores, a cada iteração calcula-se a diferença d entre a menor e a segunda menor credibilidades (linha 11). Para o usuário de menor credibilidade calcula-se uma alocação parcial como sendo o menor valor dentre a quantidade de recursos disponíveis r, o número de recursos requisitados pelo usuário $\boldsymbol{a}[i]$, e a diferença calculada d (linha 13). Essa forma evita não só que mais recursos sejam alocados do que disponíveis, mas também que o usuário não receba mais recursos do que precisa. Se a alocação parcial ainda não satisfizer as necessidades do usuário, sua credibilidade descontada pela alocação parcial é reinserida na fila (linha 19). Para o caso em que a oferta é maior que a demanda, é possível usar o mesmo algoritmo. Bastando inverter o sinal das entradas ($\boldsymbol{a} = [-a_1, \ldots, -a_N], \boldsymbol{c} = [-c_1(t), \ldots, -c_N(t)]$) e da saída ($\boldsymbol{o} = [-o_1, \ldots, -o_N]$). Para o caso excepcional em que que a demanda é igual à oferta, o algoritmo não é necessário, bastando fazer $\boldsymbol{o} = \boldsymbol{a}$.

[2]Por definição o somatório das alocações em Ψ é nulo e por consequência as credibilidades também.

Algoritmo 1 Alocação de recursos entre os usuários (demanda maior que a oferta).

Precondições: $\boldsymbol{a} \in \mathcal{A}$, $\boldsymbol{c} = [c_1(t), \ldots, c_N(t)]$
1: **para** $i \leftarrow 1, N$ **faça**
2: **se** $\boldsymbol{a}[i] > 0$ **então**
3: Inserir$(Q, \langle \boldsymbol{c}[i], i \rangle)$ ▷ Insere na fila Q as credibilidades dos usuários que precisam de recursos.
4: $\boldsymbol{o}[i] \leftarrow 0$
5: **senão**
6: $\boldsymbol{o}[i] \leftarrow \boldsymbol{a}[i]$ ▷ Todos os recursos oferecidos são alocados.
7: $r \leftarrow -\sum_{i=1}^{N} \min(\boldsymbol{a}[i], 0)$
8: **enquanto** $r > 0$ **faça** ▷ Enquanto há recursos disponíveis.
9: $\langle u, i \rangle \leftarrow \text{Extrair}_{\text{Min}}(Q)$ ▷ u é a melhor credibilidade.
10: $\langle v, j \rangle \leftarrow \text{Encontrar}_{\text{Min}}(Q)$ ▷ v é a segunda melhor credibilidade.
11: $d \leftarrow \lceil v - u \rceil$
12: **se** $d = 0$ **então** d $\leftarrow 1$
13: $\alpha \leftarrow \min(r, \boldsymbol{a}[i], d)$ ▷ Calcula uma alocação de recursos parcial.
14: $\boldsymbol{o}[i] \leftarrow \boldsymbol{o}[i] + \alpha$
15: $r \leftarrow r - \alpha$
16: **se** $\alpha < \boldsymbol{a}[i]$ **então** ▷ Usuário ainda precisa de mais recursos.
17: $u \leftarrow u + \alpha$
18: $\boldsymbol{a}[i] \leftarrow \boldsymbol{a}[i] - \alpha$
19: Inserir$(Q, \langle u, i \rangle)$ ▷ Reinsere o usuário na fila.
20: **retorne** $\boldsymbol{o}$

5. Avaliação Teórica

Esta seção mostra como o mecanismo satisfaz os requisitos introduzidos na Seção 2. Contudo, devido a restrições de espaço, as demonstrações foram deixadas para o relatório técnico [Sadok et al., 2016].

O requisito de Racionalidade Individual faz com que nenhum usuário obtenha vantagens ao deixar o mecanismo. Da forma como a função utilidade foi definida, isso equivale a garantir que nenhum usuário tenha utilidade negativa.

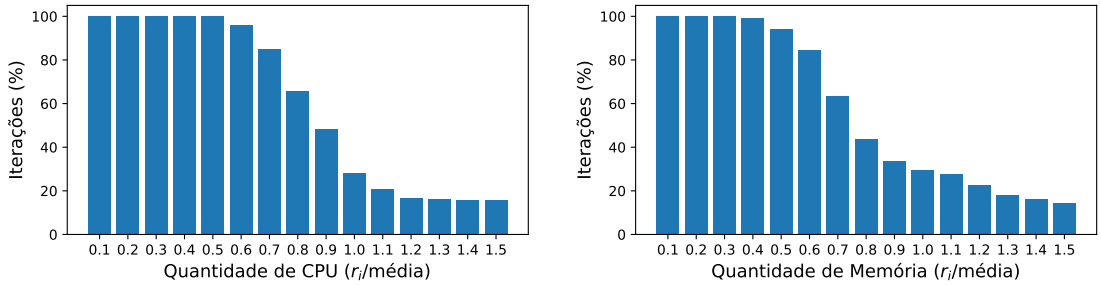
Teorema 1. ***Racionalidade Individual.*** *O mecanismo $(\mathcal{A}, \mathcal{M})$, em que $\mathcal{M}$ produz uma saída no conjunto Ψ (Equação 3), satisfaz a Racionalidade Individual. Ou seja, para todo i e $\boldsymbol{\theta}$, $u_i(\mathcal{M}(\boldsymbol{\theta}), \theta_i) \geq 0$.*

Para que o mecanismo satisfaça a Eficiência Estrita de Pareto é necessário que a alocação de recursos gere a maior soma de utilidades possível e que os usuários anunciem a verdade.

Teorema 2. ***Eficiência Estrita de Pareto.*** *Se o mecanismo $(\mathcal{A}, \mathcal{M})$ for verdadeiro e $\mathcal{M}$ satisfizer a restrição imposta pela Equação 5, então o mecanismo satisfaz a Eficiência Estrita de Pareto. O que significa dizer que, em equilíbrio, ele seleciona resultados $\boldsymbol{o} \in \mathcal{O}$ os quais satisfazem: $\sum_{i \in \mathcal{N}} u_i(\boldsymbol{o}, \theta_i) \geq \sum_{i \in \mathcal{N}} u_i(\boldsymbol{o}', \theta_i), \forall \boldsymbol{o}' \in \mathcal{O}$.*

O fato de o mecanismo ser verdadeiro, além de ser útil para garantir a Eficiência Estrita de Pareto, faz com que nenhum usuário consiga manipular o mecanismo. Os usuários são incentivados a falar a verdade caso eles não consigam ganhos de utilidade ao mentir.

Teorema 3. ***Verdade.*** *O mecanismo $(\mathcal{A}, \mathcal{M})$ em que $\mathcal{M}$ satisfaz a Equação 4 é verdadeiro. Ou seja, para todo i e θ_i, a estratégia no equilíbrio do usuário i é $a_i = \theta_i$.*



(a) Iterações com uso total de CPU.

(b) Iterações com uso total de memória.

Figura 3. Percentual de iterações onde o sistema se encontra sobrecarregado. Observe que, para uma parcela de até 0,5 da média de CPU ou 0,4 da média de memória, o sistema está completamente sobrecarregado.

Os demais requisitos não são definidos formalmente. Há um incentivo à colaboração uma vez que usuários com maior credibilidade recebem prioridade de alocação de recursos. Além disso, o fato da média das credibilidades dos usuários ser sempre zero faz com que os novos usuários entrem no sistema com a credibilidade igual à média, o que garante competitividade aos novos entrantes.

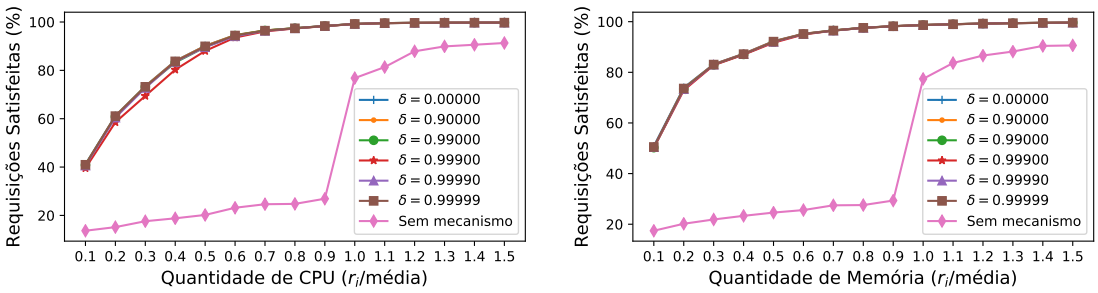
6. Avaliação Prática

Além da análise teórica, é feita uma análise com dados de uso de uma nuvem real. O conjunto de dados utilizado representa 29 dias na nuvem do Google, com aproximadamente 12.500 máquinas e 930 usuários [Reiss et al., 2011]. Os dados contêm informação das tarefas disparadas pelos usuários. Cada tarefa possui informação de início, término, identificador anônimo do usuário, além das quantidades requeridas de CPU e memória. Para as simulações é escolhido um conjunto de 100 usuários aleatórios e é usado um intervalo de 10 min entre cada iteração.[3] Além disso, uma vez que o Google normaliza a quantidade de CPU e memória requisitados por cada tarefa, esses valores são desnormalizados de forma que o menor valor presente no conjunto de dados equivalha a uma unidade do recurso. Eventuais valores racionais são arredondados para o menor inteiro maior ou igual (função teto).[4]

A partir das tarefas de cada usuário é possível inferir a quantidade de recursos que este precisa. Quando uma tarefa começa, incrementa-se a necessidade por recursos do usuário de acordo com a quantidade de recursos requeridos pela tarefa. Quando uma tarefa termina decrementa-se essa mesma quantidade. Contudo, há uma diferença importante em relação à posse de recursos: no Google, as máquinas são de propriedade da empresa, a qual possui autoridade sobre a alocação; enquanto nas nuvens colaborativas, cada usuário detém uma quantidade de recursos, hipoteticamente insuficiente para seus picos de uso. Como forma de aplicar o mesmo comportamento dos usuários ao cenário colaborativo, atribui-se a cada usuário uma quantidade de recursos. Cada usuário recebe uma quantidade de recursos proporcional à média de suas necessidades durante o período completo do conjunto de dados. Como forma de avaliar a proposta para diferentes

[3] 10 min é a duração mínima de uma tarefa na nuvem do Google.

[4] O código das simulações pode ser obtido em `https://www.gta.ufrj.br/~sadok/codes/credibility_allocation/`.



(a) Necessidades de CPU satisfeitas.

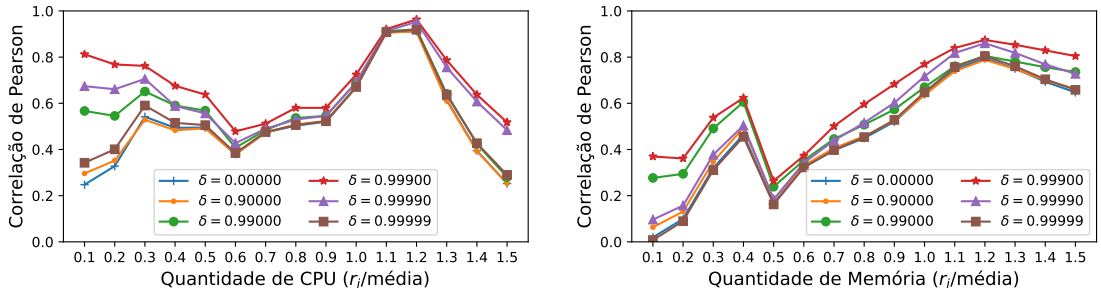
(b) Necessidades de memória satisfeitas.

Figura 4. Comparação das requisições atendidas ao usar ou não o mecanismo. O número de requisições atendidas é consideravelmente maior ao usar o mecanismo, independentemente do δ utilizado.

níveis de sobrecarga no sistema, esse fator de proporcionalidade é variado de 0,1 a 1,5 nos experimentos realizados. Quando o fator de proporcionalidade é igual a 1,0 os usuários recebem uma quantidade de recursos exatamente igual à média de suas necessidades. Uma vez definida a quantidade de recursos que cada usuário possui, os tipos dos usuários podem ser definidos. O tipo de um usuário em um determinado instante de tempo é a diferença entre a quantidade de recursos que este usuário possui e a sua necessidade por recursos nesse mesmo instante. Quando a necessidade do usuário supera os seus recursos, o tipo se torna positivo e ele pede recursos. Quando a necessidade é menor que a quantidade de recursos do usuário, o tipo se torna negativo e o usuário oferece recursos.

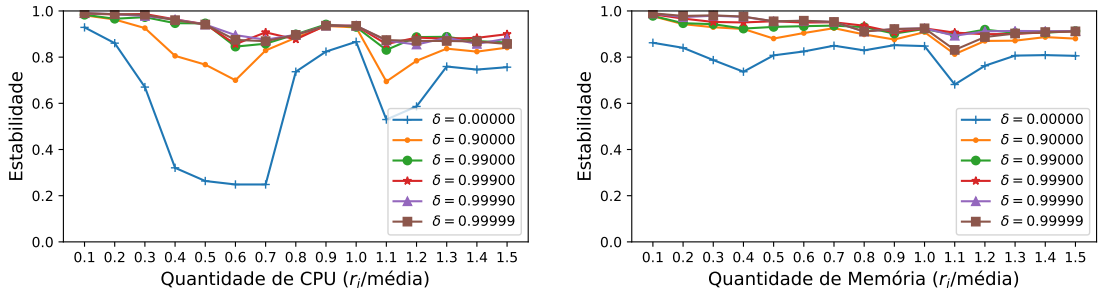
6.1. Desempenho Global

Primeiro, analisa-se o sistema para verificar o desempenho global, ou seja, qual a eficiência no uso de recursos e qual o impacto na satisfação geral dos usuários. A Figura 3 mostra a sobrecarga do sistema para quantidades diferentes de recursos atribuídos a cada usuário. Quando os usuários possuem uma quantidade de recursos até 0,5 da média de CPU, todos os recursos de CPU são usados em todas as iterações. O mesmo ocorre até 0,4 da média de memória. Já para mais de 0,5 da média de CPU e memória o sistema não se encontra mais sobrecarregado o tempo todo. Naturalmente, conforme os usuários possuem mais recursos próprios, a sobrecarga no sistema diminui; com a diminuição da sobrecarga, mais requisições por recursos são atendidas pelo sistema. A Figura 4 mostra o percentual de requisições atendidas para diferentes valores do parâmetro δ e para diferentes níveis de sobrecarga do sistema. Além disso compara-se com o percentual de requisições atendidas caso os usuários não fizessem parte do mecanismo, usando apenas os próprios recursos. O número de requisições atendidas ao usar o mecanismo é consideravelmente maior e é pouco influenciada pelo parâmetro δ. Nota-se também que, sem o mecanismo, há um aumento brusco no número de requisições satisfeitas para 1,0 da média de recursos atribuídos, tanto para CPU quanto para memória. Isso decorre do fato de muitas requisições estarem próximas da média. Assim, quando os usuários possuem mais recursos do que a média de suas necessidades, essas requisições passam a ser atendidas. No entanto, mesmo com o salto de requisições atendidas, as requisições satisfeitas sem usar o mecanismo demoram a se aproximar de 100% devido a incidência frequente de picos de demanda, os quais não podem ser atendidos sem o uso da nuvem.



(a) Correlação entre CPU contribuída e recebida. (b) Correlação entre memória contribuída e recebida.

Figura 5. Correlação de Pearson entre os recursos contribuídos e recebidos no mecanismo. Observe como $\delta = 0{,}999$ resulta na melhor correlação para todas as parcelas de recursos atribuídos.



(a) Estabilidade da alocação de CPU. (b) Estabilidade da alocação de memória.

Figura 6. Mediana das Autocorrelações *lag-1* das alocações de CPU e memória de todos os usuários.

6.2. Influência do Parâmetro δ

O valor de δ determina o quanto as alocações passadas influenciam as credibilidade dos usuários. Como foi visto, ele não influencia o número de requisições atendidas no sistema. Contudo, espera-se que valores de δ próximos de 0 façam com que os usuários não tenham suas contribuições passadas devidamente recompensadas, enquanto valores de δ muito próximos de 1 devem fazer com que a credibilidade demore a convergir e não represente as contribuições dos usuários. Os valores de δ são variados em intervalos logarítmicos já que a média móvel exponencial também é influenciada logaritmicamente. Uma forma de observar a influência do valor de δ na recompensa dos usuários é calcular o coeficiente de correlação de Pearson entre os recursos contribuídos e recebidos. Quanto mais próximo de 1 for o coeficiente, maior a correlação, o que indica a tendência dos usuários que contribuíram mais recursos também receberem mais. A Figura 5 mostra a correlação de Pearson para os diferentes valores de δ e para diferentes níveis de sobrecarga do sistema. A maior correlação, em todos os níveis de sobrecarga e para os dois tipos de recursos foi para $\delta = 0{,}999$. Como esperado, valores de δ muito próximos de 1 (como $\delta = 0{,}99999$) ou muito pequenos (como $\delta = 0$) apresentaram desempenho inferior.

O valor de δ também influencia a estabilidade das alocações. Valores de δ mais próximos de 0 fazem com que as credibilidades se alterem mais facilmente, fazendo com que as decisões de alocação do mecanismo também mudem mais facilmente. Essas osci-

lações degradam o desempenho do sistema. Para medir a influência de δ na estabilidade foi usada a autocorrelação *lag-1* ($\mathcal{R}_1$) da alocação de recursos, calculada em função das alocações $o_i(t)$ do usuário i em cada instante t: $\mathcal{R}_1(i) = \frac{\sum_{t=1}^{n-1}(o_i(t)-\bar{o_i})(o_i(t+1)-\bar{o_i})}{\sum_{t=1}^{n}(o_i(t)-\bar{o_i})^2}$, onde n é o número de iterações na duração do conjunto de dados analisado e $\bar{o_i}$ é a média das alocações do usuário i. Quanto mais próximo de 1 for $\mathcal{R}_1$, mais estáveis são as alocações. A Figura 6 compara as medianas das autocorrelações *lag-1* das alocações de todos os usuários para diferentes valores de δ e de sobrecarga do sistema. Observe como, para os menores valores de δ (0,0 e 0,9), as alocações foram menos estáveis.

7. Trabalhos Relacionados

O compartilhamento de recursos em sistemas par-a-par foi notavelmente analisado por Buragohain et al. [Buragohain et al., 2003] usando teoria dos jogos. A proposta busca aumentar a reciprocidade de contribuição de recursos entre pares de nós, considerando a falta de confiança entre os usuários do sistema. Um outro trabalho com objetivo semelhante é o FD-NoF [Falcão et al., 2015], que usa uma rede de favores para promover justiça através da contenção de recursos dos usuários de uma nuvem par-a-par. Embora a contenção de recursos auxilie na promoção de justiça, isso gera perda da eficiência, agravada ainda mais por não se considerar as reciprocidades indiretas entre os usuários. Diferentemente das propostas para sistemas par-a-par, a centralização do controle nas nuvens colaborativas permite a presença de um sistema único de reputação, onde reciprocidades indiretas podem ser consideradas. O presente trabalho usa essa vantagem, promovendo a máxima eficiência do uso dos recursos e, ao mesmo tempo, incentivando a colaboração entre os usuários do sistema.

Os trabalhos que analisam a possibilidade de coalizão entre provedores de nuvem também possuem semelhanças com este trabalho. Samaan [Samaan, 2014] criou um modelo para definir preços e a quantidade de recursos que cada provedor de nuvem deve contribuir para um mercado de excedentes (*spot market*). Também é usado um modelo com jogo repetido, mas o equilíbrio foi garantido com uma estratégia de *grim trigger*. Niyato et al. [Niyato et al., 2011] também tratam de coalizões entre provedores de nuvem mas analisam as utilidades totais para diferentes combinações de provedores, o modelo da proposta exige o uso de pagamento entre as partes para formar as coalizões. Este trabalho não assume a presença de pagamento entre as partes e propõe um equilíbrio sem usar estratégias severas como o *grim trigger*. Além disso, diferente dos outros trabalhos, a proposta é validada também usando um traço real de utilização de uma nuvem.

8. Conclusões

As nuvens colaborativas permitem que usuários com poucos recursos possam usar recursos ociosos de outros usuários. Embora os ganhos de eficiência sejam claros, é necessário a presença de um mecanismo para reger as alocações e garantir um bom equilíbrio para o sistema. Foi proposto um mecanismo que rege os pedidos dos usuários com base em suas respectivas credibilidades. As alocações de recursos privilegiam os pedidos de recursos dos usuários com as melhores credibilidades e as requisições de doação de recursos dos usuários de pior credibilidade. Foi mostrado que o sistema faz uso eficiente dos recursos, incentiva a colaboração e a honestidade dos participantes. Além disso, o sistema de credibilidade com média nula dá oportunidade de novos usuários se integrarem à nuvem. A avaliação com dados reais de utilização permitiu verificar o desempenho

da proposta para diferentes valores do parâmetro do mecanismo. Como trabalho futuro, pretende-se usar o mesmo mecanismo para outras aplicações que envolvam o compartilhamento de recursos como, por exemplo, redes elétricas inteligentes.

Referências

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. e Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50.

Buragohain, C., Agrawal, D. e Suri, S. (2003). A game theoretic framework for incentives in P2P systems. *Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003)*, 0121562:48–56.

Costa, L. H. M. K., de Amorim, M. D., Campista, M. E. M., Rubinstein, M. G., Florissi, P. e Duarte, O. C. M. B. (2012). Grandes massas de dados na nuvem: Desafios e técnicas para inovação. *Minicursos do Simpósio Brasileiro de Redes de Computadores*, p. 58.

Couto, R. S., Sciammarella, T., Barreto, H. F. S. S. M., Campista, M. E. M., Rubinstein, M. G., Costa, L. H. M. K., Vetter, F. e Marins, A. (2015). GT-PID: Uma nuvem IaaS universitária geograficamente distribuída. *TICAL2015*, p. 19.

Falcão, E. d. L., Brasileiro, F., Brito, A. e Vivas, J. L. (2015). Controlando a contenção de recursos para promover justiça em uma federação peer-to-peer de nuvens privadas. *XIII Workshop em Clouds e Aplicações (WCGA2015)*.

Myerson, R. B. (1982). Optimal coordination mechanisms in generalized principal–agent problems. *Journal of mathematical economics*, 10(1):67–81.

Niyato, D., Vasilakos, A. V. e Kun, Z. (2011). Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach. *Proceedings - 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, p. 215.

Reiss, C., Wilkes, J. e Hellerstein, J. L. (2011). Google cluster-usage traces: format + schema. Technical report, Google Inc., Mountain View, CA, USA. Revisado em 2012-03-20. `http://code.google.com/p/googleclusterdata/wiki/TraceVersion2`.

Sadok, H., Campista, M. E. M. e Costa, L. H. M. K. (2016). Um mecanismo para compartilhamento de recursos em nuvens colaborativas baseado na credibilidade dos usuários. Relatório Técnico GTA-16-47, GTA/PEE/UFRJ.

Samaan, N. (2014). A Novel Economic Sharing Model in a Federation of Selfish Cloud Providers. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):12–21.

Sotomayor, B., Montero, R. S., Llorente, I. M. e Foster, I. (2009). Virtual Infrastructure Management in Private and Hybrid Clouds. *IEEE Internet Computing*, 13(5):14–22.

Vaquero, L. M., Rodero-Merino, L., Caceres, J. e Lindner, M. (2008). A break in the clouds. *ACM SIGCOMM Computer Communication Review*, 39(1):50.

Weins, K. (2016). Cloud computing trends: 2016 state of the cloud survey. `http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2016-state-cloud-survey`. Acessado: 2016-09-19.

Trilha Principal do SBRC 2017
Sessão Técnica 12
Redes Veiculares I

Um Algoritmo de Posicionamento de Pontos de Coleta para uma Rede de Sensores Baseada em Ônibus Urbanos

Pedro Cruz[1], **Rodrigo S. Couto**[2] **e Luís Henrique M. K. Costa**[1] *

[1]Universidade Federal do Rio de Janeiro - PEE/COPPE/GTA

[2]Universidade do Estado do Rio de Janeiro - PEL/DETEL/FEN

{cruz,luish}@gta.ufrj.br, rodrigo.couto@uerj.br

Abstract. *Smart Cities can use data obtained from different sensors in order to offer better services. Such data is usually obtained by sensor networks. However, covering the whole city area with static sensors might be too expensive. This way, an alternative is to explore the strategy of embedding sensors into vehicles, enabling the mobility of the sensors and reducing the number of sensors needed. This paper analyses the performance of a network formed by buses from public transport system as a mobility platform for sensors capable of generating alert messages. This paper studies the trade-off between latency of the generated messages and number of collection points.*

Resumo. *Cidades Inteligentes podem utilizar dados obtidos por uma rede de sensores para oferecer melhores serviços. Entretanto, uma rede de sensores estáticos cobrindo toda a área de uma cidade pode apresentar um custo proibitivo. Assim, uma alternativa é explorar a estratégia de instalar sensores em veículos, que oferecem mobilidade ao sensoriamento, reduzindo o número de sensores necessários. Este trabalho analisa o desempenho de uma rede formada pelos ônibus de transporte público da cidade, como plataforma de mobilidade para sensores. O trabalho também propõe um algoritmo para a escolha de pontos de coleta e o avalia utilizando informações reais dos ônibus no Rio de Janeiro, analisando o compromisso entre a latência na entrega dos dados dos sensores e o número de pontos de coleta utilizados.*

1. Introdução

As grandes cidades acumulam problemas gerados pela alta densidade populacional. A oferta de serviços básicos à população, como transporte e segurança, é frequentemente um desafio. O conceito de Cidades Inteligentes é uma das formas de enfrentar essas dificuldades [Chourabi et al., 2012]. Dados sobre a cidade e seus serviços são coletados. A partir dessas informações, gera-se conhecimento e torna-se possível tomar decisões mais eficientes no gerenciamento da infraestrutura das cidades.

Nesse contexto, as redes de sensores servem para coletar e entregar dados que processados embasam decisões inteligentes sobre uma cidade. Tradicionalmente, os nós de uma rede de sensores sem fio coletam dados e os entregam a alguns nós especiais, denominados sorvedouros ou pontos de coleta. Os pontos de coleta são capazes de se

*Este trabalho foi realizado com recursos da FAPERJ, CNPq e CAPES.

conectar a uma rede externa, tipicamente a Internet. Através da Internet, os dados podem alcançar o nó gerenciador de operação, que é o destino final dos dados do ponto de vista da rede de sensores e de outro lado fornece serviços aos usuários finais a partir dos dados coletados [Akyildiz et al., 2002]. De acordo com a arquitetura de Internet das Coisas para Cidades Inteligentes proposta em [Zanella et al., 2014], o gerenciador de operação deve ser uma plataforma de computação em nuvem, capaz de fornecer elasticidade e resiliência aos serviços oferecidos.

A granularidade espacial de dados coletados com sensores estáticos é proporcional à quantidade de sensores envolvidos na coleta [Liu et al., 2005]. Para coletar dados por toda a área de uma Cidade Inteligente, podem ser necessários muitos sensores. Além disso, a rede utilizada para coletar os dados provenientes dos sensores também deve abranger toda a cidade. Por conta desses fatores, a solução pode não ser escalável, pois o custo de instalar e manter uma rede de sensores espalhada por toda uma cidade pode superar os benefícios proporcionados por tal projeto [Hancke et al., 2012]. Para aumentar a cobertura de sensores, [Liu et al., 2005] propõem mover os sensores pela área que se deseja sensoriar. Além de aumentar a área coberta, a movimentação dos nós também explora a entrega oportunística dos dados, como mostrado em [Ekici et al., 2006]. Dessa forma, não há a necessidade de uma rede cobrindo toda a cidade.

No entanto, a utilização de sensores móveis traz desafios, como estimar o atraso na entrega das mensagens. Como os sensores não estão em contato com pontos de coleta por todo o tempo, um sensor que possua dados a serem entregues deve aguardar até o próximo contato com algum ponto de coleta. Esse atraso pode restringir a utilização dos dados por parte de algumas aplicações. Tome-se como exemplo a emissão de um alerta de enchentes para a população. Nessa aplicação, sensores de chuva emitem alertas caso os índices pluviométricos instantâneos excedam um determinado limiar. Caso os dados sobre chuvas intensas demorem horas para alcançar a aplicação, os alertas serão inúteis para que se evite os transtornos causados por enchentes. Por isso, é importante que o atraso entre a coleta de um dado e sua entrega a um ponto de coleta seja estimado. Assim, é possível definir as aplicações que podem utilizar os dados coletados por essa rede.

Apesar da vasta literatura em sensoriamento móvel e redes oportunísticas, o estado da arte foca principalmente na mobilidade dos pontos de coleta, sem analisar os efeitos do número e do posicionamento dos pontos de coleta no atraso da entrega dos dados [Zhao et al., 2016]. Este trabalho visa a utilização dos ônibus do transporte público de uma cidade como plataforma de mobilidade para os nós de sensoriamento, com pontos de coleta posicionados nos pontos de ônibus. O trabalho analisa a utilização de sensores embarcados nos ônibus do sistema de transporte público para entregar dados em uma aplicação de emissão de alertas. Para tal, propõe-se um algoritmo de escolha de pontos de coleta. Assim, este trabalho apresenta três contribuições principais:

- a caracterização do atraso da entrega dos dados em uma rede de sensores embarcados nos ônibus de uma cidade, utilizando dados reais de tráfego de ônibus no Rio de Janeiro;
- a formulação de um problema de programação linear inteira (*Integer Linear Programming* - ILP) para escolha dos pontos de coleta, visando minimizar o atraso de entrega dos dados coletados;
- a proposta de um algoritmo guloso para a escolha dos pontos de coleta, que possui

resultados próximos à solução ótima do ILP proposto.

Este trabalho está estruturado da seguinte forma. A Seção 2 explica o cenário de sensoriamento explorado. A Seção 3 descreve os trabalhos relacionados, enquanto a Seção 4 apresenta o modelo de rede de sensores utilizado. A Seção 5 apresenta a formulação de uma solução ótima para o problema abordado. A Seção 6 propõe um algoritmo para obter uma solução sub-ótima em tempo viável e compara seus resultados com a solução ótima. A Seção 7 aplica o algoritmo proposto na rede formada pelos ônibus e pontos de parada da cidade do Rio de Janeiro, utilizando dados reais de seus posicionamentos. Por fim, a Seção 8 conclui o trabalho e apresenta direções futuras.

2. Sensoriamento Urbano Utilizando Ônibus de Transporte Público

Este trabalho aborda a utilização dos ônibus do transporte público para fornecer mobilidade aos sensores de uma cidade, com baixo atraso na entrega dos dados. A utilização dos ônibus como plataforma de mobilidade tem como vantagem o fato dos ônibus possuírem trajetórias previsíveis, serem parte da infraestrutura pública e não ser necessário gasto de energia com a mobilidade.

No cenário estudado, cada ônibus carrega um nó de sensoriamento, capaz de realizar medidas, armazená-las e transmiti-las através de uma interface sem fio para um ponto de coleta. Os pontos de coleta, localizados em pontos de ônibus, enviam pela Internet os dados recebidos para o nó gerenciador de operação. O gerenciador de operação, executado como um serviço de computação em nuvem, provê serviços para os usuários finais. O esquema está ilustrado na Figura 1.

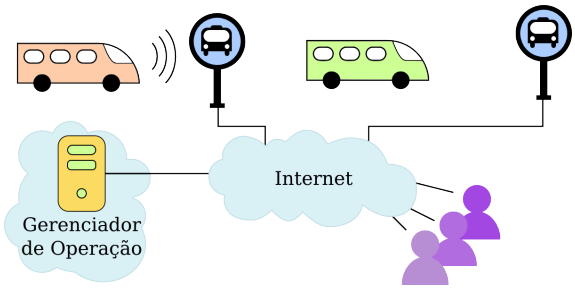


Figura 1. O esquema de distribuição através de ônibus.

A arquitetura de cada nó móvel de sensoriamento, proposta anteriormente em [Cruz et al., 2016], está ilustrada na Figura 2. Cada nó possui um Controlador, que gerencia todas as funções de coletar, armazenar temporariamente e transmitir os dados. O Receptor do Sistema Global de Navegação via Satélite (*Global Navigation Satellite System* - GNSS) calcula o posicionamento do nó e o Banco de Sensores coleta dados ambientais. A interface sem fio possibilita a conexão do nó com os pontos de coleta, para que os dados sensoriados possam ser transmitidos. A unidade Memória Persistente permite que os dados coletados sejam armazenados. Dessa forma, caso não haja uma conexão com um ponto de coleta, os dados coletados podem ser armazenados para envio posterior. Por último, a Fonte de Energia fornece alimentação ao nó, podendo ser alimentada pelo próprio ônibus. A uma determinada taxa de amostragem, o Controlador consulta os dados coletados pelo Banco de Sensores e os dados de posicionamento fornecidos pelo

Receptor GNSS. O Controlador armazena na unidade de Memória Persistente esses dados juntamente com o momento da coleta.

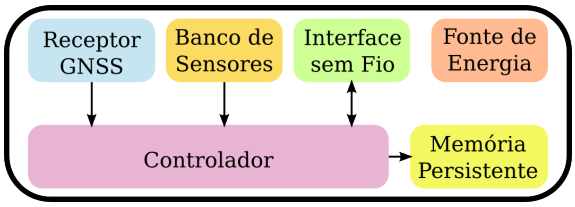


Figura 2. A arquitetura de um nó de sensoriamento embarcado em um ônibus.

Assume-se que os pontos de coleta são a única forma de contato de um nó com a nuvem. Em um trajeto, o intervalo de tempo d que o nó percorre entre dois pontos de coleta consiste no tempo durante o qual o nó não pode enviar dados à nuvem. Assim, é importante que esse intervalo d seja compatível com as aplicações que utilizam os dados sensoriados. Por exemplo, uma aplicação que coleta o histórico de medidas de temperatura da cidade, para posterior análise, pode tolerar atrasos da ordem de horas ou até dias. Por outro lado, as aplicações de alerta descritas anteriormente, como notificações de enchentes, podem tolerar atrasos de apenas alguns minutos.

A rede estudada é uma rede de sensores móveis sem fio. Essas redes possuem topologia dinâmica [Romer e Mattern, 2004], de forma que dois nós da rede podem estar conectados em alguns momentos e desconectados em outros. Por isso, nem sempre existe um caminho entre um sensor qualquer da rede e um ponto de coleta.

Neste trabalho, considera-se que não existe encaminhamento de dados *entre* nós de sensoriamento. O nó que coleta um determinado conjunto de dados é responsável por entregar esse conjunto de dados a um ponto de coleta. Assim, o atraso de envio após um dado ter sido sensoriado por um nó é igual ao tempo desde o sensoriamento do dado até o encontro do nó com um ponto de coleta, mais o tempo de transmissão do dado do nó até o ponto de coleta, mais o tempo de transmissão do ponto de coleta até a nuvem.

Tipicamente, os tempos de encaminhamento do nó até o ponto de coleta e de transmissão do ponto de coleta até o gerenciador de operação são da ordem de milissegundos. Além disso, o tempo que um ônibus demora desde o último encontro com um ponto de ônibus até um próximo encontro com um ponto de ônibus é da ordem de minutos. Por causa dessa discrepância na ordem de grandeza dos valores, este trabalho considera desprezível o atraso de encaminhamento do nó ao ponto de coleta e do ponto de coleta até a nuvem. Apenas o atraso gerado pela espera por conexão é considerado.

3. Trabalhos Relacionados

A literatura sobre posicionamento de pontos de coleta em redes de sensores sem fio estáticos é bem vasta e abrange a otimização de diversas métricas de qualidade de serviço. Wong *et al.* propõem um algoritmo que busca a menor latência para uma rede de sensores estáticos, a partir da configuração espacial de seus pontos de coleta [Wong et al., 2004]. O trabalho propõe um algoritmo computacionalmente custoso capaz de obter a solução ótima e um algoritmo menos custoso, capaz de obter uma aproximação para a configuração de seus pontos de coleta. Uma vez que o tempo de

armazenamento e encaminhamento de um conjunto de dados é muito maior do que o tempo de propagação dos dados em um enlace, a métrica utilizada por Wong *et al.* para minimizar o atraso nas mensagens é o número de saltos. Este trabalho diferencia-se de [Wong et al., 2004] pois são utilizadas redes dinâmicas, formada por ônibus, objetivando reduzir os custos de implantação, mas mantendo a área coberta pelos sensores.

Existem outros trabalhos que estudam o sensoriamento de Cidades Inteligentes através de sensores embarcados em veículos urbanos. O projeto Opensense utiliza sensores embarcados em veículos do transporte público para monitorar partículas de poluição dispersas no ar [Marjovi et al., 2015]. Esse projeto utiliza a rede GSM para a entrega de dados, sem explorar a comunicação oportunística possibilitada pela mobilidade dos nós que compõem a rede. O projeto Mosaic segue uma abordagem semelhante, utilizando sensores embarcados em veículos para medir partículas de poluição [Dong et al., 2015]. Para tal, utiliza WiFi, GPRS ou Bluetooth como tecnologias para a entrega de mensagens aos pontos de acesso. O foco desses trabalhos é o tratamento de dados para sensores de poluição na presença de mobilidade, sem a análise da entrega das medidas obtidas pelos sensores, foco do presente trabalho.

O projeto BusNet [Zoysa et al., 2007] utiliza os ônibus do transporte público para obter dados sobre poluição e também para monitorar as condições da pavimentação das ruas. Porém, a abordagem não procura estabelecer a latência no tempo de entrega das medidas aos usuários e às aplicações. Dessa forma, o trabalho é limitado a aplicações tolerantes a atrasos indefinidos na entrega dos dados. Em contraste, este trabalho quantifica o atraso que os dados devem sofrer desde sua coleta até sua entrega.

[Dias e Costa, 2016] analisam a vazão de uma rede na qual os ônibus de transporte público podem se comunicar e mostram que existe potencial de transferência de dados suficientes para diversos tipos de aplicações, inclusive sensoriamento. O presente trabalho procura caracterizar o atraso na entrega dos dados, a fim de definir melhor as aplicações em sensoriamento que podem utilizar os ônibus enquanto plataforma de mobilidade.

4. Modelo de Rede de Sensores com Pontos de Coleta Limitados

Neste trabalho, assume-se que, dado o custo de implantação, o número de pontos de coleta na rede de sensores móveis formada pelos ônibus deve ser limitado. Assim, o problema é decidir quais pontos de ônibus devem ser escolhidos como pontos de coleta da rede analisada. Esta seção descreve o modelo de redes proposto. A Tabela 1 fornece as notações utilizadas.

Seja $b \in \mathcal{B}$ um ônibus que coleta dados, $p \in \mathcal{P}$ um ponto (parada) de ônibus candidato a ponto de coleta, P_b uma lista ordenada na qual cada elemento de $P_b(i)$ é o i-ésimo ($1 \leq i \leq m$) ponto p com o qual o ônibus b teve contato, e $T_b(i)$ o instante no qual o ônibus b teve contato com o i-ésimo ponto de P_b. É possível interpretar a sequência P_b como o trajeto de b entre os pontos com os quais b tem contato.

Quando um ônibus $b \in \mathcal{B}$ coleta dados ao longo de P_b, esses dados sofrem atrasos de, no máximo, $(T_b(2) - T_b(1), T_b(3) - T_b(2), ..., T_b(m) - T_b(m-1))$. O atraso $T_b(2) - T_b(1)$ é experimentado apenas pelos dados coletados imediatamente após b perder o contato com $P_b(1)$. Todos os outros dados coletados pelo ônibus no caminho entre $P_b(1)$ e $P_b(2)$ sofrem atrasos menores, até que b encontre $P_b(2)$. O mesmo raciocínio é

Tabela 1. Notações utilizadas no trabalho.

Notação	Descrição	Tipo
$\mathcal{B}$	Ônibus que circulam pela cidade	Conjunto
$\mathcal{P}$	Candidatos a ponto de coleta da cidade	Conjunto
$\mathcal{P}_{bp}^{s}$	Candidatos a ponto de coleta que fazem contato com o ônibus b anteriormente ao ponto p	Conjunto
$\mathcal{P}_{bp}^{d}$	Candidatos a ponto de coleta que fazem contato com o ônibus b posteriormente ao ponto p	Conjunto
$\mathcal{I}$	Pontos de ônibus que são os primeiros ou os últimos pontos de qualquer um dos ônibus	Conjunto
P_b	A sequência de pontos de ônibus que fazem contato com o ônibus b, por ordem de contato	Parâmetro
$P_b(i)$	Função que retorna o i-ésimo elemento da sequência P_b	Parâmetro
$T_b(i)$	Instante no qual o ônibus b encontra o i-ésimo elemento da sequência P_b	Parâmetro
D_b	A sequência de intervalos entre contatos do ônibus b com pontos de coleta para o ônibus b	Parâmetro
$D_b(i)$	O i-ésimo elemento de D_b e o intervalo de tempo entre o contato com $P_b(i)$ e o contato com $P_b(i+1)$	Parâmetro
d_{bpq}	Atraso total, para um ônibus b, entre os pontos p e q	Parâmetro
$N_{desejado}$	Quantidade desejada de pontos de coleta	Parâmetro
$A_{máx}$	Maior atraso possível na rede entre dois pontos de um ônibus	Variável
x_i	Valor binário que indica se o ponto i é escolhido como ponto de coleta	Variável
y_{bpq}	Valor binário indicando, para um ônibus b, se q é o próximo ponto de coleta a partir do ponto p	Variável

válido para qualquer par $(P_b(i), P_b(i+1))$ do trajeto P_b. Por motivos de simplicidade, neste trabalho $(T_b(i+1) - T_b(i))$ é denominado o atraso entre $P_b(i)$ e $P_b(i+1)$.

É possível, então, definir D_b como a sequência de atrasos para cada ônibus b e obter os valores de cada elemento em D_b a partir da subtração:

$$D_b = \{T_b(2) - T_b(1), T_b(3) - T_b(2), ..., T_b(m) - T_b(m-1)\}. \quad (1)$$

Pode-se, então, definir o atraso máximo da rede como sendo o maior atraso nas sequências de atrasos de todos os ônibus:

$$A_{máx} = \max_{b \in \mathcal{B}}(\max(D_b)). \quad (2)$$

4.1. Remoção de um candidato a ponto de coleta

No problema estudado, todos os pontos de ônibus $p \in \mathcal{P}$ são candidatos a pontos de coleta. Se o ponto de ônibus $p \in \mathcal{P}$ não for escolhido para ser ponto de coleta, diz-se que o ponto p foi removido. Quando um ponto p é removido, os nós que poderiam descarregar dados neste ponto de coleta agora devem descarregar no próximo ponto de seu trajeto que seja um ponto de coleta. Os nós que podem descarregar em p são todos os nós b que possuem p em uma ou mais posições de sua sequência P_b.

A remoção de p possui efeitos na sequência D_b, ilustrados na Figura 3. Se o ponto p é o elemento $P_b(i)$ e p é removido, então o os dados coletados a partir de $P_b(i-1)$ somente são entregues quando o nó tem contato com o ponto de coleta subsequente, $P_b(i+1)$. Assim, o atraso do $(i-1)$-ésimo elemento em P_b, denotado por $D_b(i-1)$, é acrescido de $D_b(i)$. Além disso, $D_b(i)$ é removido. Em virtude da remoção de $P_b(i)$ e $D_b(i)$ das sequências P_b e D_b, todos os pontos posteriores a $P_b(i)$ e $D_b(i)$ são deslocados uma posição para trás.

A remoção de um candidato a ponto de coleta não possui os mesmos efeitos para ônibus diferentes, pois os ônibus podem seguir trajetos diferentes. A Figura 4 ilustra um exemplo do efeito da remoção do candidato q para dois ônibus, $b, c \in \mathcal{B}$ que se locomovem pelos pontos $p, q, r, s, t \in \mathcal{P}$. A sequência de pontos de b é $P_b = (p, q, r)$ e a sequência de

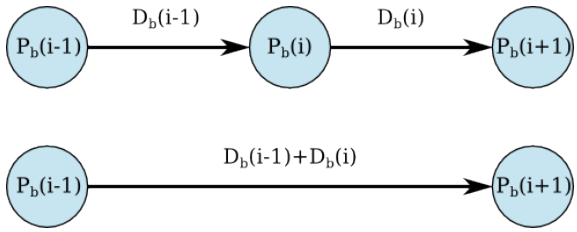


Figura 3. O efeito no tempo entre contatos causado pela remoção de um candidato a ponto de coleta.

atrasos é $D_b = (1, 2)$. A sequência de pontos de c é $P_c = (s, q, t)$ e a sequência de atrasos é $D_c = (3, 1)$. Quando ocorre a remoção do candidato a ponto de coleta q, P_b se torna (p, r), D_b se torna (3), P_c se torna (s, t) e D_c se torna (4).

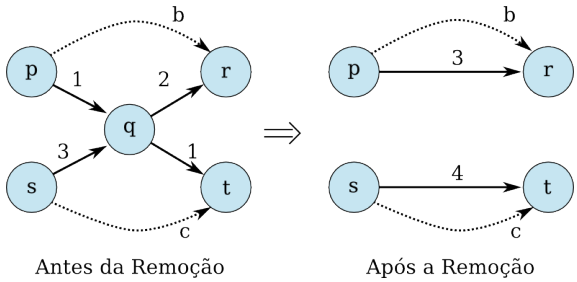


Figura 4. Remoção de um ponto de coleta do ponto de vista de ônibus diferentes.

Define-se, então, a operação de remoção de um candidato a ponto de coleta $p \in \mathcal{P}$:

1. Exclusão de p do conjunto $\mathcal{P}$
2. Para cada $b \in \mathcal{B}$:
   (a) Se houver algum $P_b(i) = p$, para qualquer i:
      i. Remoção $P_b(i)$ de P_b
      ii. Atribuição de $D_b(i-1) := D_b(i-1) + D_b(i)$
      iii. Remoção $D_b(i)$ de D_b

A remoção do ponto p altera um ou mais atrasos em D_b, quando p é parte de P_b. O *atraso de remoção de* p ($A_{rem}(p)$) é definido como o maior atraso que pode ser causado pela remoção de p:

$$A_{rem}(p) = \max_{b \in \mathcal{B}}(\max_{i \in \{2,\ldots,(|P_b|-1)\}} (\{D_b(i-1) + D_b(i) | P_b(i) = p\})). \quad (3)$$

De acordo com o modelo adotado, os pontos que estão no início ou no final de um trajeto P_b não podem ser removidos. O ponto inicial não pode ser removido pois a operação de remoção de um ponto de coleta não é definida para pontos que não possuam um ponto anterior. O último ponto de P_b não pode ser removido porque o último atraso em D_b é o intervalo de tempo entre o penúltimo e o último pontos de P_b. A remoção do último ponto de P_b causaria a indefinição desse atraso. Além disso, o último ponto de P_b não pode ser removido para garantir a entrega de mensagens, já que dados de sensores podem ser gerados até o final do trajeto. O conjunto dos pontos iniciais e finais de todas as sequências P_b para todos os ônibus em $\mathcal{B}$ é denotado conjunto $\mathcal{I}$.

Escolhe-se como métrica objetivo o atraso máximo da rede, pois esse será maior atraso experimentado por uma mensagem de alerta emitida por um nó embarcado em um ônibus. Existem outras medidas possíveis para escolher pontos de coleta para uma rede de sensores embarcados em ônibus do transporte público. Por exemplo, a menor soma dos atrasos esperados, definida por $A_{SUM} = \sum_{b \in B} \sum_{d \in D_b} d$. Essa medida garante que, na média, o atraso experimentado pelas mensagens será o menor possível. Porém, um algoritmo que busque o menor atraso possível no caso médio, não necessariamente minimiza os atrasos considerados grandes. É possível que algumas mensagens tenham atrasos esperados inaceitáveis. Assume-se uma aplicação de emissão de alertas onde todas as mensagens são consideraras críticas. Nessa hipótese, é importante garantir que todas as mensagens sejam entregues em tempo hábil. Baseado no modelo descrito, este trabalho propõe uma heurística para minimizar o aumento do atraso causado pela remoção de um candidato a ponto de coleta. A Seção 5 formula uma solução ótima para o problema, porém computacionalmente intensa, enquanto a Seção 6 apresenta um algoritmo guloso baseado na heurística de remoção do candidato a ponto de coleta com menor atraso de remoção, dado pela Equação 3.

5. Formulação da Solução Ótima

O problema deste trabalho escolhe, dentre todos os pontos existentes, uma quantidade $N_{desejado}$ de pontos de coleta. Essa escolha busca minimizar o maior atraso possível na rede entre dois pontos de coleta, definido na Equação 2. A solução ótima para esse problema é obtida através da formulação de um problema de programação linear inteira (ILP), como se segue:

$$\text{minimizar } A_{máx} \tag{4}$$

$$\text{sujeito a } \sum_{p \in \mathcal{P}} x_p = N_{desejado}; \tag{5}$$

$$\sum_{q \in \mathcal{P}^d_{bp}} y_{bpq} = x_p \quad \forall b \in B, \quad \forall p \in \mathcal{P}^s_b; \tag{6}$$

$$\sum_{p \in \mathcal{P}^s_{bq}} y_{bpq} = x_q \quad \forall b \in B, \quad \forall q \in \mathcal{P}^d_b; \tag{7}$$

$$A_{máx} - \sum_{q \in \mathcal{P}^d_{bp}} d_{bpq} y_{bpq} \geq 0 \quad \forall b \in B, \quad \forall p \in \mathcal{P}^s_b; \tag{8}$$

$$x_p = 1 \quad \forall p \in \mathcal{I} = \left(\left(\bigcup_{b \in B} \mathcal{P}^s_b \setminus \mathcal{P}^d_b \right) \bigcup \left(\bigcup_{b \in B} \mathcal{P}^d_b \setminus \mathcal{P}^s_b \right) \right); \tag{9}$$

$$A_{máx} \in \mathbb{Z}; \quad y_{bij} \in \{0,1\} \; \forall \, b \in \mathcal{B}, \; \forall \, i,j \in \mathcal{P}; \quad x_i \in \{0,1\} \; \forall \, i \in \mathcal{P}. \tag{10}$$

O objetivo do problema, dado pela Equação 4, é minimizar $A_{máx}$. A Equação 5 especifica que a quantidade de pontos de coleta escolhidos deve ser igual a $N_{desejado}$. A Equação 6 indica que, se o ponto i for escolhido e se o ônibus b faz contato com algum outro ponto após i, então i deve ser antecessor de algum ponto j. Analogamente, a

Equação 7 indica que, se o ponto j for escolhido e se o ônibus b faz contato com algum outro ponto antes de j, então j deve ser sucessor de algum ponto i. Assim, as Equações 6 e 7 juntas definem as variáveis y_{bij} para o cálculo do atraso mostrado na Figura 3. Isto é, se em um trajeto do ônibus b os pontos i e j forem utilizados e os outros pontos entre eles não, o atraso entre esses dois pontos será utilizado para o cálculo de $A_{máx}$ na Equação 8. A Equação 9 especifica que, se um ponto i é o primeiro ou o último ponto que qualquer um dos ônibus faz contato, então i é obrigatoriamente escolhido como ponto de coleta, como visto na Seção 4.1. Finalmente, a Equação 10 define o domínio de cada variável.

6. Algoritmo de Escolha dos Pontos de Coleta

A solução do problema ótimo é NP-difícil, por se tratar de um ILP, e assim não escala. Dessa forma, propõe-se um algoritmo guloso que inicializa considerando que todos os pontos de ônibus candidatos a ponto de coleta são pontos de coleta. A cada iteração, o algoritmo remove do conjunto dos candidatos o ponto com menor atraso de remoção. A operação de remoção de um candidato a ponto de coleta é definida na Seção 4.1. O algoritmo termina quando o conjunto solução possui $N_{desejado}$ elementos ou quando todos os candidatos devem ser pontos de coleta, pois $|\mathcal{P}| - |\mathcal{I}| <= N_{desejado}$.

O algoritmo recebe como parâmetros: o conjunto $\mathcal{B}$; o conjunto $\mathcal{P}$; o conjunto $\mathcal{I}$; o vetor $\boldsymbol{trajetos}$, que contém uma lista $\boldsymbol{P_b}$ para cada elemento em $b \in \mathcal{B}$ ($\boldsymbol{P_{bi}}$ é a sequência de candidatos a ponto de coleta pelos quais b_i passa); o vetor $\boldsymbol{d_{prox}}$, que contém, em cada índice i a sequência de atrasos $\boldsymbol{D_b}$ para b; e $N_{desejado}$, que é o número de pontos que se deseja manter.

Algoritmo 1 Algoritmo de Escolha dos Pontos de Coleta

Precondições: $\mathcal{B} = \{b_1, \ldots, b_K\}, \mathcal{P} = \{p_1, \ldots, p_N\}, \mathcal{I} = \{p_{py}, \ldots, p_{pz}\}, \boldsymbol{trajetos} = (\boldsymbol{P_{b1}}, \ldots, \boldsymbol{P_{bK}}), \boldsymbol{d\_prox} = (\boldsymbol{D_{b1}}, \ldots, \boldsymbol{D_{bK}}), N_{desejado}$

se $|\mathcal{P}| - |\mathcal{I}| \leq N_{desejado}$ **então retorne** $\mathcal{I}$ ▷ Não é possível obter $N_{desejado}$ pontos
para $b \in \mathcal{B}$ **faça** ▷ Inicializar a lista de atrasos de remoção
  para $i \leftarrow 1, |\boldsymbol{P_b}|$ **faça**
    $\boldsymbol{max\_ts}[\boldsymbol{P_b}[i]] \leftarrow \max(\boldsymbol{D_b}[i-1] + \boldsymbol{D_b}[i], \boldsymbol{max\_ts}[\boldsymbol{P_b}[i]])$
$\mathcal{R} \leftarrow \emptyset$
enquanto $|\mathcal{R}| + |\mathcal{I}| + N_{desejado} < |\mathcal{P}|$ **faça** ▷ Remover os pontos até o número desejado
  para $p \in \mathcal{P}$ **faça** ▷ Selecionar um ponto com menor atraso de remoção, não removido nem ilícito
    se ($d\_min == NULO$ **ou** $d\_min < \boldsymbol{max\_ts}[p]$) **e** $p \notin I$ **e** $p \notin R$ **então**
      $d\_min \leftarrow \boldsymbol{max\_ts}[p]$
      $p\_a\_remover \leftarrow p$
  para $\boldsymbol{P_b} \in \boldsymbol{trajetos}$ **faça**
    para $i \leftarrow 1, |\boldsymbol{P_b}|$ **faça**
      se $\boldsymbol{P_b}[i] = p\_a\_remover$ **então**
        $\boldsymbol{D_b}[i-1] \leftarrow \boldsymbol{D_b}[i-1] + \boldsymbol{D_b}[i]$ ▷ Aumentar a espera do ponto anterior
        se $\boldsymbol{max\_ts}[\boldsymbol{P_b}[i+1]] < \boldsymbol{D_b}[i-1] + \boldsymbol{D_b}[i]$) **então**
          $\boldsymbol{max\_ts}[\boldsymbol{P_b}[i+1]] \leftarrow \boldsymbol{D_b}[i-1] + \boldsymbol{D_b}[i])$ ▷ Atualizar atrasos de remoção
        se $i > 1$ **então**
          se ($\boldsymbol{max\_ts}[\boldsymbol{P_b}[i-1]] < \boldsymbol{D_b}[i-2] + \boldsymbol{D_b}[i-1]$) **então**
            $\boldsymbol{max\_ts}[\boldsymbol{P_b}[i-1] \leftarrow \boldsymbol{D_b}[i-2] + \boldsymbol{D_b}[i-1]$
        Remover($\boldsymbol{P_b}[i]$) ▷ Remover o elemento i de $\boldsymbol{P_{bi}}$
  Inserir($\mathcal{R}, p$) ▷ Inserir o ponto escolhido no conjunto de pontos removidos
retorne $\{\mathcal{S} \setminus \mathcal{R}\}$

No Algoritmo 1, o vetor $\boldsymbol{max\_ts}$ mantém, para cada ponto p, seu atraso de remoção. A função Remover($\boldsymbol{P_{bi}}, p$), na linha 20, remove o ponto p da lista $\boldsymbol{P_{bi}}$.

O laço de repetição da linha 6 é o laço principal do algoritmo. A cada iteração, um ponto é escolhido para ser removido. A escolha do ponto a ser removido, entre as linhas 7 e 10, escolhe um ponto $p\_a\_remover$ que tenha o menor atraso de remoção que não esteja nem no conjunto $\mathcal{I}$ dos pontos ilícitos e nem no conjunto $\mathcal{R}$, dos pontos já removidos. A atualização dos atrasos máximos do ponto anterior e do ponto posterior a p acontece entre as linhas 15 e 19. Na linha 20, o ponto p é removido de P_b.

O algoritmo retorna o conjunto $\{\mathcal{S} \setminus \mathcal{R}\}$ de pontos de coleta escolhidos quando o conjunto possui tantos elementos quanto $|\mathcal{P}| - N_{desejado}$, ou retorna um conjunto vazio quando não é possível remover nenhum ponto, pois o número de elementos no conjunto $\mathcal{I}$ de pontos ilícitos é maior do que $|\mathcal{S}| - N_{desejado}$.

6.1. Análise da complexidade de tempo do algoritmo

As entradas do Algoritmo 1 são o conjunto $\mathcal{B}$, de tamanho K; o conjunto $\mathcal{P}$, de tamanho N; o vetor de listas $\boldsymbol{trajetos}$, de tamanho K; o vetor de listas $\boldsymbol{d\_prox}$, de tamanho K; e o inteiro $N_{desejado}$. Seja M o tamanho da maior lista em $\boldsymbol{trajetos}$. Consequentemente, a maior lista em $\boldsymbol{d\_prox}$ possui tamanho $M - 1$.

A complexidade de tempo do algoritmo é a complexidade do teste inicial, na linha 1, somada à complexidade da inicialização da lista de atrasos por remoção, na linha 2, somada à complexidade da remoção dos pontos, na linha 6.

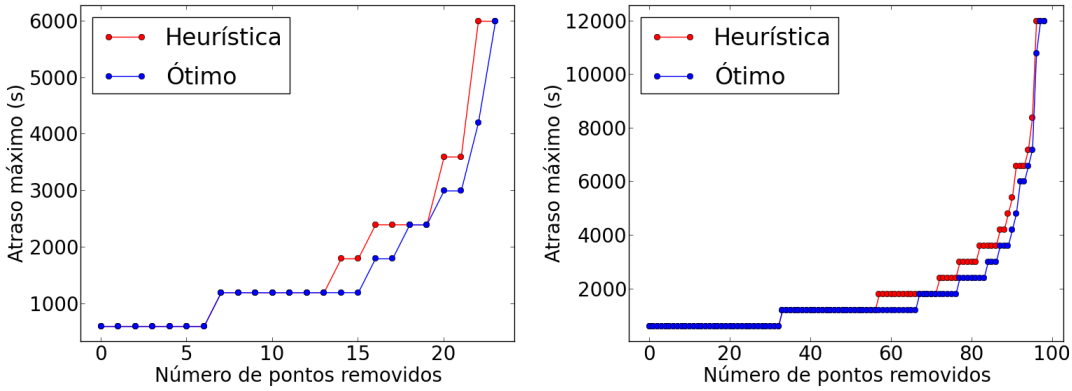
A verificação inicial envolve comparar a cardinalidade dos conjuntos $\mathcal{P}$ e $\mathcal{I}$ com $N_{desejado}$. Portanto, utilizando a notação O para análise de pior caso, a complexidade desse trecho é $O(1)$.

A operação de inicialização das listas de espera máxima de remoção envolve uma iteração sobre todos os M elementos de $\boldsymbol{P_b}$ aninhada em uma iteração sobre todos os K elementos de $\mathcal{B}$. Dessa forma, a complexidade desse trecho é $O(KM)$.

A remoção dos pontos possui um laço externo, na linha 6, no qual $|\mathcal{R}|$ cresce de uma unidade a cada iteração até, no máximo, atingir o valor de $N_{desejado}$. Assim, a complexidade desse laço é a complexidade dos laços internos multiplicada por N, que é a cardinalidade de pior caso para $\mathcal{R}$.

Aninhados ao laço da linha 6 existem dois outros laços, em sequência. Na linha 7, um laço percorre todos os N elementos de $\mathcal{P}$, verificando se $p \in \mathcal{I}$ e se $p \in \mathcal{R}$. Sabendo que os conjuntos $\mathcal{I}$ e $\mathcal{R}$ são sempre subconjuntos de $\mathcal{P}$, é possível utilizar estruturas de dados nas quais a complexidade das verificações $p \in \mathcal{R}$ e $p \in \mathcal{I}$ seja $O(1)$. Logo, a complexidade do laço da linha 7 é $O(N)$.

O laço da linha 20 percorre todos os K elementos P_b da lista $trajetos$. Aninhado ao laço da linha 20, um outro laço itera sobre todos os, no máximo, M elementos de cada P_b. Durante cada iteração, o elemento D_b do vetor $\boldsymbol{max\_ts}$ é acessado e os elementos $D_b(i-2)$, $D_b(i-1)$ e $D_b(i)$ são acessados. Para evitar percorrer toda a lista D_b até o elemento i, é possível utilizar um ponteiro em $P_b(i)$ que aponte para $D_b(i)$. Dessa forma, o acesso a $D_b(i-2)$, $D_b(i-1)$ e $D_b(i)$ dentro dessa iteração tem complexidade de pior caso $O(1)$. Assim, o laço da linha 20 possui complexidade $O(KM)$.



(a) Grade de 25 pontos, com 5 ônibus. (b) Grade de 100 pontos, com 10 ônibus.

Figura 5. Maior atraso da rede para número de pontos removidos em uma grade 25 pontos e em uma grade 100 pontos.

A função $Inserir(\mathcal{R}, p)$ insere p no conjunto $\mathcal{R}$ e, portanto, tem complexidade $O(1)$. Dadas as complexidades dos laços internos, a complexidade do laço da linha 6 é $O(N^2 + KMN + N)$.

A complexidade de tempo do algoritmo é dada por $O(1+KM+N^2+KMN+N)$. Pela notação O, a complexidade de tempo é dada pelo maior entre $O(N^2)$ e $O(KMN)$.

6.2. Comparação com o problema ótimo

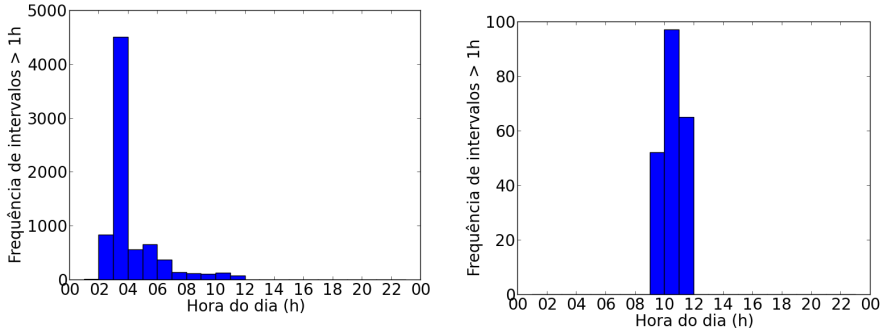
Para estimar a aproximação fornecida pelo algoritmo proposto, é realizada uma comparação entre as soluções encontradas pelo algoritmo e as soluções do problema ótimo, descrito na Seção 5. São utilizados dados sintéticos para simular um cenário de mobilidade de ônibus entre pontos de ônibus. Os pontos de ônibus são organizados em grade e é simulada a movimentação dos ônibus entre os pontos. Todos os ônibus iniciam e terminam seus trajetos no mesmo ponto. Para solução do ótimo, utiliza-se a ferramenta IBM ILOG CPLEX 12.5.1.

A Figura 5(a) mostra os resultados obtidos, para 5 ônibus percorrendo uma grade de 25 pontos, com 5 linhas e 5 colunas. Na Figura 5(b), é possível observar os resultados para 10 ônibus percorrendo uma grade de 100 pontos, com 10 linhas e 10 colunas. Pela observação dos gráficos, é possível notar que os atrasos máximos crescem em saltos, indicando que há pontos de coleta que são uma espécie de gargalo que, quando removidos, aumentam significativamente o atraso da rede. Na maior parte dos resultados, o algoritmo proposto está a menos de um gargalo de distância.

7. Utilização do Algoritmo em um Cenário Real

A Federação das Empresas de Transporte de Passageiros do Estado do Rio de Janeiro (FETRANSPOR) e a Prefeitura da Cidade do Rio de Janeiro disponibilizam dados das posições dos ônibus [IPLANRIO, 2016a] e das posições dos pontos de ônibus [IPLANRIO, 2016b] da cidade do Rio de Janeiro. A partir desses dados, gera-se as entradas do algoritmo proposto neste trabalho.

Neste trabalho, os dados são coletados por um período de 24h, desde a 0:00 h do dia 29 de novembro de 2016 até a 0:00 h do dia 30 de novembro de 2016 e inseridos em



(a) Ocorrência de atrasos maiores do que 1 h ao longo de 24 h sem filtros.

(b) Ocorrência de atrasos maiores do que 1 h ao longo de 24 h com seleção apenas de contatos entre 8:00 h e 22:00 h.

Figura 6. Distribuição de atrasos maiores que 1 h antes e após filtragem entre 8:00 h e 22:00 h.

Tabela 2. Características dos conjuntos de dados entre 8h e 22h.

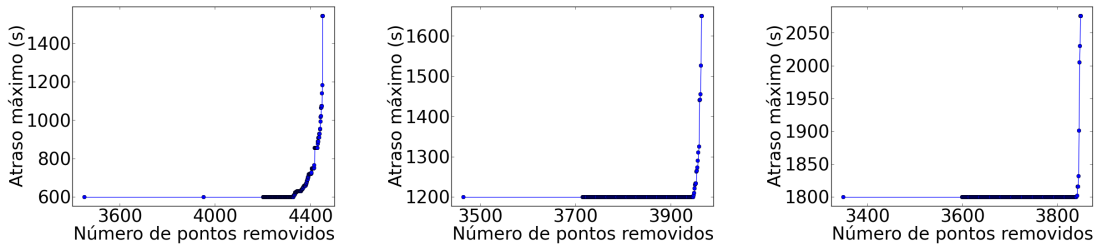
Limite de maior atraso (s)	Número de ônibus	Número de pontos	Número de pontos em $\mathcal{I}$
Sem limite	6.484	6.244	2.851
1.800	5.863	6.244	2.778
1.200	5.320	6.234	2.662
600	3.304	6.181	2.175

um banco de dados. Define-se que, caso um ônibus esteja a menos de 300 m de um ponto de ônibus, então ambos estão em contato. É selecionada a tupla (instante, ônibus, ponto de ônibus) para todos os instantes em que algum ônibus possui contato com algum ponto de ônibus. Com esses dados, são conhecidos os conjuntos $\mathcal{B}$ e $\mathcal{P}$. Para cada ônibus, são criados P_b e D_b. São obtidos dados de 6.309 pontos de ônibus e 6.627 ônibus.

Os conjuntos $\mathcal{B}$, $\mathcal{P}$ são obtidos e as listas P_b e D_b são calculadas para cada $b \in \mathcal{B}$. O gráfico da Figura 6(a) mostra a distribuição dos atrasos maiores do que 1 h ao longo das horas do dia. É possível notar um elevado número de intervalos maiores do que 1 h no período anterior às 8:00 h. Ao selecionar apenas pontos entre 8:00 h e 22:00 h é obtido o gráfico de intervalos maiores do que 1 h ilustrado na Figura 6(b). As escalas dos gráfico são diferentes, pois, do contrário, não seria possível observar os valores de ambas as figuras. Supõe-se que esses intervalos mais longos ocorram durante a madrugada pois muitos ônibus estão parados na garagem durante o período noturno.

Para eliminar o problema exposto na Figura 6(a), utiliza-se neste trabalho apenas os dados coletados entre 8:00 h e 22:00 h. Além disso, realiza-se uma nova filtragem, eliminando os ônibus que possuem algum atraso maior do que 600 s, 1.200 s ou 1.800 s. Assim, remove-se, para cada filtragem, todos os ônibus $b \in \mathcal{P}$ que não fazem contato com nenhum ponto de ônibus em um intervalo maior que o atraso máximo tolerado. Por fim, o conjunto $\mathcal{I}$ é construído. A Tabela 2 mostra características dos dados após a filtragem.

Para reduzir a influência de ruídos nos dados, utilizam-se valores médios de atraso. Assim, o atraso médio entre dois pontos $p, q \in \mathcal{P}$ é calculado a partir dos intervalos de todos os ônibus que passam pelos dois pontos em sequência. Após isso, o atraso médio é substituído em todas as sequências D_b de todos os ônibus $b \in \mathcal{B}$.



(a) Apenas ônibus sem nenhum atraso inicial maior do que 600s.

(b) Apenas ônibus sem nenhum atraso inicial maior do que 1200s.

(c) Apenas ônibus sem nenhum atraso inicial maior do que 1800s.

Figura 7. O atraso máximo da rede ($A_{máx}$) em função do número de pontos de coleta removidos.

O resultado da execução do algoritmo para o conjunto de dados coletados pode ser observado na Figura 7. O eixo X de cada gráfico representa o número de pontos removidos pelo algoritmo e o eixo Y, o atraso máximo da rede. É possível notar que o atraso máximo da rede também cresce em saltos, como visto na Seção 5.

Nos três casos avaliados, são removidos mais de 55% dos pontos de ônibus sem nenhum prejuízo ao atraso máximo inicial da rede. Dessa forma, é possível atender a uma aplicação cuja tolerância a atrasos seja de até 1.800 s, 1.200 s ou 600 s instalando pontos de coleta em apenas 45% dos pontos de ônibus da cidade.

8. Conclusões e Trabalhos Futuros

As redes de sensores sem fio são uma importante ferramenta para as Cidades Inteligentes, mas o custo de implementar esse tipo de rede por uma grande área urbana é um desafio. Para tal, utilizam-se redes de sensores sem fio móveis, que podem reduzir os custos de sensoriamento e de entrega de mensagens, através da entrega oportunística de mensagens. Este trabalho explorou a mobilidade dos ônibus do transporte público de uma cidade para conferir mobilidade a nós de sensoriamento, que entregam os dados obtidos e mensagens de alerta a pontos de coleta instalados nos pontos de ônibus.

Para análise do desempenho da rede abordada, este trabalho propôs um algoritmo de distribuição dos pontos de coleta pelos pontos de ônibus. Esse algoritmo tem como objetivo minimizar o atraso máximo da rede estudada, para um dado número de pontos utilizados. O algoritmo foi comparado com a solução ótima, disponibilizada por uma formulação de programação linear inteira, obtendo resultados próximos ao ótimo. Além disso, o algoritmo mostrou que é possível obter uma rede de coleta utilizando 49% dos ônibus da cidade do Rio de Janeiro com atraso máximo esperado de 600 s, utilizando 1.917 pontos de coleta, cerca de 30% do total de pontos de ônibus da cidade.

Como trabalhos futuros, pretende-se combinar uma métrica de minimização de atraso médio com a métrica utilizada, de forma a reduzir o atraso médio na entrega das mensagens, sem prejuízos ao atraso máximo. Além disso, serão utilizadas medidas coletadas por um período maior e novos filtros para eliminação de ruídos.

Referências

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y. e Cayirci, E. (2002). A survey on sensor networks. *IEEE communications magazine*, 40(8):102–114.

Chourabi, H., Nam, T., Walker, S., Gil-Garcia, J. R., Mellouli, S., Nahon, K., Pardo, T. A. e Scholl, H. J. (2012). Understanding smart cities: An integrative framework. Em *System Science (HICSS), 2012 45th Hawaii International Conference on*, p. 2289–2297. IEEE.

Cruz, P., Neto, J. B. P., Campista, M. E. M. e Costa, L. H. M. K. (2016). On the accuracy of data sensing in the presence of mobility. Em *7th International Conference Network of the Future*. NoF.

Dias, D. e Costa, L. H. M. K. (2016). Análise da capacidade de dados de uma rede de Ônibus urbanos. Em *XXXIV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*. SBrT.

Dong, W., Guan, G., Chen, Y., Guo, K. e Gao, Y. (2015). Mosaic: Towards city scale sensing with mobile sensor networks. Em *Parallel and Distributed Systems (ICPADS), 2015 IEEE 21st International Conference on*, p. 29–36. IEEE.

Ekici, E., Gu, Y. e Bozdag, D. (2006). Mobility-based communication in wireless sensor networks. *IEEE Communications Magazine*, 44(7):56 – 62.

Hancke, G. P., Hancke Jr, G. P. et al. (2012). The role of advanced sensing in smart cities. *Sensors*, 13(1):393–425.

IPLANRIO (2016a). Descrição do dataset conjunto gps ônibus. Disponível em `http://data.rio/dataset/gps-de-onibus`.

IPLANRIO (2016b). Documentação de paradas das linhas de ônibus. Disponível em `http://data.rio/dataset/pontos-de-parada-de-onibus`.

Liu, B., Brass, P., Dousse, O., Nain, P. e Towsley, D. (2005). Mobility improves coverage of sensor networks. Em *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*. MobiHoc.

Marjovi, A., Arfire, A. e Martinoli, A. (2015). High resolution air pollution maps in urban environments using mobile sensor networks. Em *2015 International Conference on Distributed Computing in Sensor Systems*, p. 11 – 20. IEEE.

Romer, K. e Mattern, F. (2004). The design space of wireless sensor networks. *IEEE wireless communications*, 11(6):54–61.

Wong, J. L., Jafari, R. e Potkonjak, M. (2004). Gateway placement for latency and energy efficient data aggregation [wireless sensor networks]. Em *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, p. 490–497. IEEE.

Zanella, A., Bui, N., Castellani, A., Vangelista, L. e Zorzi, M. (2014). Internet of things for smart cities. *IEEE Internet of Things Journal*, 1(1):22–32.

Zhao, D., Ma, H., Li, Q. e Tang, S. (2016). A unified delay analysis framework for opportunistic data collection. *Wireless Networks*, p. 1–13.

Zoysa, K. D., Keppitiyagama, C., Seneviratne, G. P. e Shihan, W. W. A. T. (2007). A public transport system based sensor network for road surface condition monitoring. Em *NSDR '07 Proceedings of the 2007 workshop on Networked systems for developing regions*. NSDR.

Geo-SDVN: Um Protocolo *Geocast* para Redes Veiculares Definidas por *Software*

Roniel Soares de Sousa[1], **Antonio A. F. Loureiro**[1], **Luiz Filipe M. Vieira**[1], **André C. B. Soares**[2], **Felipe Saraiva da Costa**[2]

[1] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)
CEP 31270-901 – Belo Horizonte – MG – Brasil

[2]Departamento de Computação
Universidade Federal do Piauí (UFPI)
CEP 64.049-550 – Teresina – PI – Brasil

{ronisds,loureiro,lfvieira}@dcc.ufmg.br,

andre.soares@ufpi.edu.br,felipesaraivac@outlook.com

Abstract. *Vehicular Networks represent an emerging technology that enables the use of network services by drivers and vehicle passengers. However, the heterogeneity of vehicular networks, which can use various wireless technologies, represents a challenge for the development of communication protocols. The Software Defined Vehicular Networks (SDVN) paradigm allows flexible design of these new protocols. This work proposes a new geocast protocol for Software Defined Vehicular Networks. Simulation results reveal that the proposed protocol exhibits superior results in terms of delivery rate and signaling overhead when compared to the Flooding algorithm.*

Resumo. *As Redes Veiculares representam uma tecnologia emergente que possibilita a utilização de serviços de rede por parte dos motoristas e passageiros de veículos. Porém, a heterogeneidade das redes veiculares, que podem utilizar diversas tecnologias de redes sem fio, representa um desafio para o desenvolvimento de protocolos de comunicação. O paradigma das Redes Veiculares Definidas por Software (SDVN) possibilita uma flexibilidade no desenvolvimento destes novos protocolos. Este artigo propõe um novo protocolo de geocast para Redes Veiculares Definidas por Software. Foi realizado um estudo de avaliação de desempenho no qual o protocolo proposto apresentou resultados superiores em termos de taxa de entrega e overhead de sinalização quando comparado com o algoritmo de inundação.*

1. Introdução

As Redes Veiculares (*Vehicular Ad-hoc Networks* - VANET) são sistemas de comunicação formados principalmente por veículos. Tais redes consistem em veículos automotores que se comunicam através de redes sem fio com outros veículos ou com infraestruturas localizadas à margem das vias. Essas redes são um dos componentes que formam os Sistemas Inteligentes de Transportes (*Intelligent Transportation Systems* – ITS) [dos S. Alves et al. 2009].

Dentre as características do ambiente veicular que acarretam em desafios para as VANETs estão: (i) a alta mobilidade dos nós, (ii) as mudanças constantes na topologia da rede, (iii) a existência de redes densas que ocasionam em altas taxas de perdas de pacotes por congestionamento na rede e (iv) a existência de redes esparsas que dificultam a disseminação de informações [dos S. Alves et al. 2009].

Recentemente, alguns autores sugerem e defendem a utilização de SDN (*Software Defined Networks*) no ambiente veicular. Em [He et al. 2016], é adotada a nomenclatura SDVN (*Software Defined Vehicular Network*) para as redes veiculares que utilizam a abordagem SDN. Além disso, também são propostas arquiteturas e serviços SDVN em [Kazmi et al. 2016], [Ku et al. 2014] e [He et al. 2016]. Redes Definidas por *Software* são atrativas por serem programáveis por meio de um controlador que mantém o estado da rede e podem se adaptar à mobilidade dos nós.

Uma importante classe de comunicação em redes veiculares é o *geocast*. *Geocast* é o nome dado à disseminação de mensagens para veículos dentro de uma região definida geograficamente [Maihofer 2004]. Nos últimos anos, diferentes protocolos *geocast* foram propostos para redes veiculares tradicionais [Bachir and Benslimane 2003] [Joshi et al. 2007] [Rahbar et al. 2010], sendo cada um otimizado para um cenário específico.

Este trabalho tem foco no problema de disseminação de informações em SDVN. Neste contexto, é proposto um novo protocolo *geocast* para Redes Veiculares Definidas por *Software*. O protocolo faz uso da tecnologia LTE (*Long Term Evolution*) para otimizar o roteamento, sendo utilizado principalmente no tráfego de mensagens de controle. Isso permite que mais veículos dentro da região desejada sejam alcançados.

O protocolo proposto utiliza-se das vantagens das redes SDVN para fornecer uma abordagem adaptativa. Ao utilizar o LTE, em conjunto com a tecnologia WAVE (*Wireless Access in Vehicular Environments*), para transmissão de mensagens de *geocast*, permite-se que mais veículos estejam ao alcance do remetente, ao mesmo tempo em que o número de mensagens enviadas é reduzido.

O restante deste artigo está organizado como segue. A Seção 2 apresenta alguns trabalhos relacionados. A Seção 3 detalha o funcionamento do protocolo proposto. Os resultados do estudo de avaliação de desempenho são discutidos na Seção 4 e, por fim, a Seção 5 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Aplicações de ITS podem requerer que uma mensagem seja entregue apenas para veículos situados em uma determinada região [Zhang et al. 2014]. A transmissão realizada para uma área específica é chamada de *Geocast* ou *Geobroadcast*. Neste âmbito, diferentes soluções foram propostas na última década.

Por exemplo, em [Bachir and Benslimane 2003] foi proposto um protocolo *geocast* chamado IVG (*Inter-Vehicular Geocast*). O propósito do IVG é informar veículos de uma região de risco, chamada de grupo *multicast*, sobre qualquer perigo em rodovias. O protocolo envia mensagens alertando os demais veículos da região de risco no momento em que o veículo sofre um acidente. Os veículos descartam a mensagem caso já tenham recebido. Caso contrário, eles esperam um tempo para retransmiti-la. O veículo com

menor tempo é escolhido e retransmite a mensagem.

Já em [Joshi et al. 2007] foi proposto o protocolo DRG (*Directed Robust Geocast*), que é baseado em um esquema que usa distância como suporte para realizar inundação (*flooding*) direcionada e restrita. No DRG, o veículo que origina o *broadcast* envia a mensagem para todos os vizinhos e, o tempo para retransmissão é calculado com base na distância do receptor ao remetente. Quanto maior a distância, menor o tempo. O veículo mais distante então retransmite a mensagem e, os outros veículos recebem a mensagem e cancelam sua retransmissão.

Alguns trabalhos recentes focam no aperfeiçoamento de protocolos existentes por meio de técnicas como: redução no número de mensagens trocadas utilizando como parâmetro a relação sinal ruído [Voicu et al. 2014]; técnicas de controle de acesso ao meio [Omar et al. 2013]; soluções com baixo custo energético [Kumar et al. 2016].

Como a maioria dos protocolos são otimizados para cenários específicos [He et al. 2016], alguns autores sugerem o uso de SDN em conjunto com as redes veiculares [He et al. 2016] [Ku et al. 2014] [Kazmi et al. 2016]. As redes veiculares que utilizam SDN vem sendo chamadas de SDVN e, têm como principal atrativo, a sua adaptabilidade, permitindo que a rede se adeque facilmente à mudança de topologia. Nesses trabalhos são propostas arquiteturas e serviços para SDVN.

Por ser um tema relativamente novo, existem poucos trabalhos abordando estratégias de *geocast* em SDVN. Em [Liu et al. 2015], foi proposta uma arquitetura SDVN e um protocolo *geocast* que funciona utilizando um conjunto de Unidades de Acostamento (*Roadside Units* - RSUs). Destaca-se que o conceito de SDVN é utilizado apenas nas comunicações realizadas entre as infraestruturas externas, e não entre os veículos. O protocolo proposto neste trabalho, apresentado na Seção 3, independe da existência de RSUs. Até o momento da escrita deste artigo não foram encontrados outros trabalhos que estudam a disseminação de informações em SDVN.

3. Protocolo *Geocast* para Redes Veiculares Definidas por *Software*

Nesta seção é apresentada a arquitetura SDVN utilizada, bem como os detalhes da implementação do protocolo *geocast* proposto tanto nos veículos quanto no controlador SDVN.

3.1. Visão Geral da Arquitetura

A Figura 1 descreve a arquitetura utilizada. Nessa arquitetura é utilizado um **controlador SDVN** centralizado. Os veículos utilizam a arquitetura WAVE para realizar a troca de dados entre eles. A comunicação entre o plano de dados e o plano de controle é realizada por meio da rede celular (padrão LTE), na qual os veículos trocam informações com o controlador utilizando o protocolo *OpenFlow* [McKeown et al. 2008].

Nos cenários em que a quantidade de veículos sobrecarregue o controlador centralizado, como pode acontecer em grandes cidades, poderá ser utilizada uma implementação distribuída do controlador. Para isso é necessário utilizar um componente que informa qual controlador os veículos em uma região devem utilizar. Já nos casos em que a comunicação com o controlador for interrompida, pode ser utilizado algum outro protocolo *geocast* V2V para disseminar as informações.

De forma geral, o protocolo *OpenFlow* funciona da seguinte maneira. Cada *switch OpenFlow* possui um tabela de fluxos (*flow table*) que armazena qual ação deve ser realizada com cada fluxo de dados conhecido. Cada entrada da *flow table* (*flow entry*) contém um conjunto de campos de correspondência, que identificam o fluxo, e um campo de ação, que informa a ação que deve ser realizada ao receber um pacote daquele fluxo. Quando um *switch* recebe um pacote, é verificado se existe uma entrada na *flow table* correspondente a aquele pacote. Caso exista, a ação correspondente ao fluxo é realizada. Caso contrário, o *switch* envia para o controlador uma mensagem do tipo *Table-miss*. O controlador então responde informando a ação necessária para aquele tipo de fluxo. Ao receber a mensagem de resposta, o *switch* realiza a ação informada e adiciona esta informação na sua *flow table*. Assim, quando for necessário encaminhar outro pacote correspondente ao mesmo fluxo, o *switch* já saberá qual ação deverá realizar.

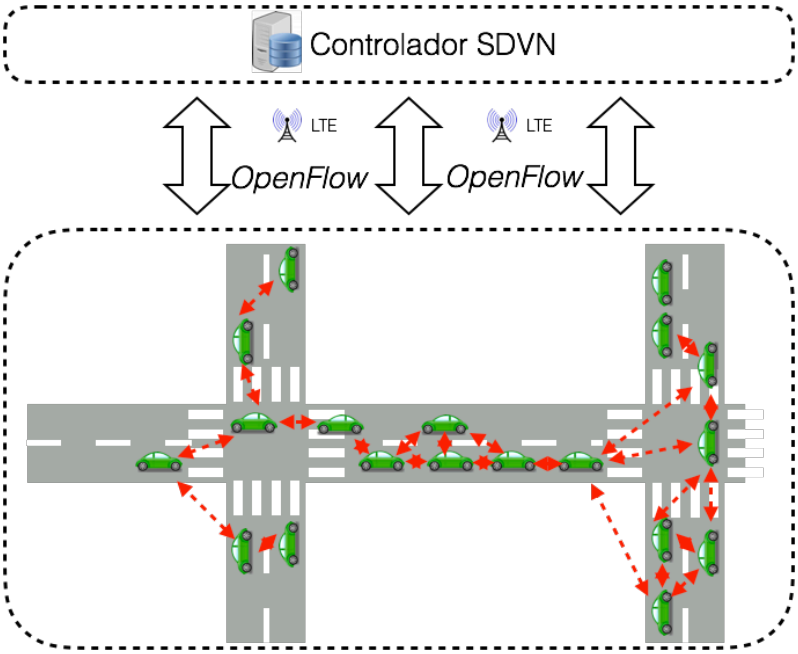


Figura 1. Arquitetura SDVN utilizada.

3.2. Implementação do Protocolo *Geocast* nos Veículos

Neste trabalho, cada veículo representa um *switch OpenFlow* e, com isso, possui uma *flow table*. A *flow table* possui dois campos de correspondência, o *Vehicle ID* e o *Geocast ID*. O campo *Vehicle ID* identifica o veículo que gerou a mensagem que deve ser transmitida por meio do protocolo *geocast*. Já o campo *Geocast ID* é um identificador da mensagem. A sua utilização é da seguinte forma. Sempre que uma aplicação deseja transmitir uma mensagem *geocast* para uma região *R*, será criado um identificador único *Geocast ID* para aquela mensagem. Se em algum momento uma dada aplicação desejar transmitir uma mensagem na mesma região R, será utilizado o mesmo identificador *Geocast ID* criado anteriormente.

A *flow table* possui ainda um campo *Timestamp*, que armazena o momento em que a *flow-entry* foi inserida na *flow table*. O campo *Timestamp* é utilizado para remover uma *flow-entry* quando esta passar um tempo determinado (*Flow-Entry_Lifetime*) sem ser atualizada.

Por fim, a *flow table* possui dois campos que identificam a ação a ser realizada com os pacotes do fluxo. São eles o campo *Actions* e o campo *Next_Hop_IDs*. O campo *Actions* possui um dos seguintes valores:

- ***DROP***: O veículo deve descartar o pacote.
- ***SEND_TO_CONTROLLER***: O veículo deve encaminhar o pacote de dados para o controlador utilizando a rede LTE.
- ***BROADCAST***: O veículo deve encaminhar o pacote para os seus veículos vizinhos utilizando *broadcast one hop*.
- ***SEND_TO_CONTROLLER+BROADCAST***: O veículo deve realizar as ações *SEND_TO_CONTROLLER* e *BROADCAST*.

Já o campo *Next_Hop_IDs* informa quais veículos vizinhos também irão retransmitir a mensagem. Esse campo é utilizado para que o veículo verifique se todos os vizinhos que devem retransmitir a mensagem de fato a retransmitiram. Isso é necessário porque se algum dos vizinhos não retransmitir a mensagem, toda a disseminação pode ficar comprometida. Assim, o veículo que está encaminhando o pacote no momento esperará um tempo *Max_Waiting_Time*. Se ao final deste tempo ele não receber as retransmissões de todos os *Next_Hop_IDs*, o veículo então transmite novamente a mensagem. Isto é realizado no máximo *Max_Tries* vezes. A Tabela 1 exemplifica uma *flow table* com quatro *flow-entries*, uma para cada ação possível.

Vehicle ID	***Geocast ID***	***Timestamp***	***Actions***	***NextHopIDs***
Veh_1	5	22.5	DROP	Empty
Veh_32	1	14.1	SEND_TO_CONTROLLER	Empty
Veh_12	2	52.8	BROADCAST	Veh_1, Veh_15
Veh_15	15	5.0	SEND_TO_CONTROLLER+BROADCAST	Veh_1, Veh_32

Tabela 1. Exemplo de uma *flow table*.

Como discutido anteriormente, quando um veículo não possui uma entrada na *flow table* correspondente ao pacote que deseja transmitir, este enviará ao controlador uma mensagem do tipo *Table-miss*. Uma mensagem *Table-miss* possui os seguintes campos:

- ***Vehicle ID***: Identificador do veículo que criou a mensagem.
- ***Geocast ID***: Identificador *geocast* da mensagem, que é criado como discutido anteriormente.
- ***Region Center***: Coordenadas que identificam o centro da região de interesse da mensagem.
- ***Region Radius***: Raio da região de interesse da mensagem.

Ao enviar uma *Table-miss*, o veículo esperará *Table-miss_Waiting_Time* segundos. Ao final deste tempo, o veículo verifica se foi recebida uma mensagem de resposta *Table-miss_Response* do controlador. Caso positivo, a ação presente na resposta é realizada. Caso contrário, o veículo pode esperar novamente por *Table-miss_Waiting_Time* segundos uma resposta. Esta espera acontece no máximo *Table-miss_Max_Waiting_Tries* vezes. Ou seja, o veículo esperará a resposta por no máximo *Table-miss_Waiting_Time* $\times$ *Table-miss_Max_Waiting_Tries* segundos. Se ao final deste tempo não for obtida uma resposta, o veículo envia novamente uma mensagem *Table-miss* ao controlador. Acontecem no máximo *Table-miss_Max_Tries* tentativas de envio da *Table-miss* ao controlador. Se não for obtida a resposta do controlador, o veículo simplesmente descarta o pacote.

A mensagem de resposta do controlador, chamada de *Table-miss_Response*, conterá todos os campos presentes na mensagem *Table-miss*, mais os campos abaixo:

- ***Actions***: Ação que deve ser realizada (*DROP*, *BROADCAST*, etc).
- ***Next_Hop_IDs***: Lista de identificadores dos próximos saltos, como discutido anteriormente. Essa lista é vazia caso a ação não seja do tipo *BROADCAST* ou *SEND_TO_CONTROLLER+BROADCAST*.

3.3. Controlador SDVN

O controlador SDVN é utilizado para informar aos veículos quais ações devem ser realizadas para cada fluxo de dados. A escolha das ações que cada veículo deve realizar para determinado fluxo de dados é feita da seguinte forma. Inicialmente, é necessário que o controlador possua informação global da topologia da rede veicular. Para isto, todos os veículos enviam *beacons* contendo suas posições geográficas obtidas por GPS ao controlador com uma determinada frequência (por exemplo, um *beacon* por segundo). O controlador mantém uma tabela *Vehicle_Table* com a posição de cada veículo. A Tabela 2 apresenta os campos presentes na *Vehicle_Table*.

Vehicle ID	***Position***	***Timestamp***
Veh_1	(15.3, 210.1, 0)	52.1
Veh_32	(258.0, 1330.1, 0)	52.5
Veh_12	(1530.8, 10.1, 0)	52.7
Veh_15	(72.1, 28.1, 0)	53.0

Tabela 2. Tabela *Vehicle_Table* armazenada no controlador com as posições dos veículos.

O campo *Vehicle ID* é o identificador único do veículo. O campo *Position* possui a posição presente no último *beacon* recebido do veículo representado por *Vehicle ID*. Já o *Timestamp* representa o momento em que foi recebido o último *beacon* do veículo *Vehicle ID*. Caso se passem *Vehicle_Entry_Duration* segundos desde o recebimento do último *beacon* de um determinado veículo, a sua entrada é removida da *Vehicle_Table*.

A partir da *Vehicle_Table*, o controlador constrói uma matriz de adjacência correspondente ao estado atual de conexão dos veículos na rede veicular. Para isso, o controlador possui a informação da potência de transmissão dos veículos. A partir da potência de transmissão dos veículos, o controlador calcula qual o valor aproximado do raio de comunicação *Transmission_Range* de todos os veículos, em metros. O controlador assume que dois veículos podem comunicar um com o outro se a distância entre eles for menor ou igual a *Transmission_Range* menos um valor de segurança igual a *Guard_Range* metros. O *Guard_Range* é utilizado para minimizar os erros provocados por perdas de pacotes.

Sempre que o controlador recebe uma *Table-miss*, ele decidirá as ações que todos os veículos da região de interesse devem realizar da seguinte forma. Primeiro é verificado se a região de interesse da mensagem referente ao *Table-miss* recebido está conectada. Ou seja, é verificado se é possível alcançar todos os veículos a partir do veículo que criou a mensagem, utilizando apenas a comunicação V2V. Caso positivo, os seguintes passos são executados. O primeiro é calcular o *Minimum Connected Dominating Set - MCDS* (Conjunto Dominante Conectado Mínimo) do grafo formado pelos veículos presentes na região. Porém, sabe-se que o MCDS é um problema *NP-Hard* tanto para grafos arbitrários não direcionados quanto para grafos do tipo *Unit Disk* [Clark et al. 1990]

[Guha and Khuller 1998]. Assim, faz-se necessária a utilização de heurísticas para calcular soluções aproximadas de instâncias do problema MCDS em tempo factível. Neste trabalho, foi utilizada a heurística proposta em [Butenko et al. 2004]. Essa heurística foi utilizada por ser simples de ser implementada e apresentar bons resultados. Destaca-se que é realizada uma pequena modificação na heurística para que o veículo que criou a mensagem, a ser transmitida em *geocast*, sempre faça parte do MCDS calculado.

Os veículos escolhidos como pertencentes ao conjunto dominante conectado serão os veículos responsáveis por retransmitir a mensagem. Os demais veículos irão apenas descartar a mensagem.

O controlador calcula os *Next_Hop_IDs* de cada veículo que deverá retransmitir a mensagem da seguinte forma. Cria-se um subgrafo formado apenas pelos veículos que deverão retransmitir a mensagem, ou seja, aqueles contidos no MCDS. Após isso, é realizada uma busca em profundidade a partir do veículo que criou a mensagem. Durante a busca em profundidade, quando visita-se um determinado nó, todos os seus vizinhos que ainda não foram visitados são adicionados à sua lista de *Next_Hop_IDs*. O fato de utilizar uma busca em profundidade no subgrafo formado pelos veículos contidos no MCDS faz com que o tamanho do conjunto *Next_Hop_IDs* de cada veículo seja pequeno.

Assim, após calcular o MCDS aproximado e as listas de *Next_Hop_IDs*, o controlador enviará uma mensagem do tipo *Table-miss_Response* para cada veículo presente na região de interesse. Os veículos que estão no conjunto dominante receberão uma *Table-miss_Response* com a ação *BROADCAST* e seus respectivos *Next_Hop_IDs*, e os demais veículos receberão uma *Table-miss_Response* com a ação *DROP*.

Como discutido anteriormente, no momento em que o veículo que originou a mensagem verificar que acabou o tempo correspondente ao *Table-miss_Waiting_Time*, este executará a ação presente na *Table-miss_Response* recebida. Ao receberem o pacote enviado pelo veículo que originou a mensagem, os demais veículos contidos no MCDS irão retransmiti-lo, pois estes já terão recebido mensagens *Table-miss_Response* do controlador.

Com isso, o controlador consegue tomar as decisões quando a região de interesse da mensagem está conectada. Em cenários onde a densidade de veículos da região de interesse é baixa, pode acontecer da região em questão não ser conectada. Neste caso, o controlador detectará cada sub-região conectada dentro da região de interesse, e executará os mesmos passos utilizados quando a região é conectada para cada uma dessas sub-regiões conectadas. O controlador também escolherá um veículo de cada sub-região para agir como o veículo que originou a mensagem, ou seja, para começar a disseminação da mensagem na sua sub-região conectada. Este veículo é chamado de *Region_Head*. Já o veículo que originou a mensagem receberá uma *Table-miss_Response* contendo a ação *BROADCAST+SEND_TO_CONTROLLER*, diferente do que acontece no cenário em que a região de interesse é conectada. Isso se faz necessário para que sempre que o veículo decidir enviar uma mensagem na região, este transmita a mensagem para seus veículos vizinhos por meio do *BROADCAST* e também para o controlador por meio da rede LTE. Ao receber a mensagem, o controlador irá retransmiti-la para todos os *Region_Heads*, que, por sua vez, irão começar a disseminação da mensagem nas sub-regiões.

O controlador armazena em uma tabela todas as mensagens *Table-miss* recebi-

das nos últimos *Table-miss_Time* segundos. A cada *Update_Table-miss_Entry* segundos o controlador verificará se precisa atualizar as ações de algum dos veículos que estão na região referente àquela mensagem *Table-miss*. Qualquer modificação em alguma das ações criadas anteriormente faz com que o controlador envie uma *Table-miss_Response* ao veículo correspondente informando a modificação. Essas modificações ocorrem devido à mobilidade dos veículos que faz com que a matriz de adjacência altere constantemente.

4. Avaliação de Desempenho

Foi realizado um estudo de avaliação de desempenho por meio de simulações. O objetivo é verificar a eficácia do protolo *geocast* em disseminar informações. As principais métricas de avaliação utilizadas são: (i) taxa de entrega de mensagens, (ii) *overhead* de sinalização e (iii) atraso na entrega de conteúdo.

4.1. Cenário

O desempenho do protocolo foi comparado com um algoritmo *geocast* simples de *Flooding* no qual todos os veículos na região de interesse retransmitem a mensagem uma vez. Apesar de não ser um protocolo de roteamento *geocast*, o algoritmo de *geocast Flooding* é útil para comparação com outros protocolos *geocast*. Isso se deve ao fato do mesmo ser utilizado como bloco de construção para diversos protocolos de roteamento [Maihofer 2004].

As ferramentas de simulação utilizadas foram o simulador de eventos discretos OMNeT++ 4.6 [Varga and Hornig 2008] em conjunto com o simulador de tráfego e mobilidade urbana SUMO 0.25.0 [Krajzewicz et al. 2012]. O *framework* Veins LTE 1.3 [Hagenauer et al. 2014], que é utilizado no estudo de redes veiculares heterogêneas e possui suporte para o padrão IEEE 802.11p, foi utilizado para integrar esses dois simuladores.

Por meio do simulador de tráfego SUMO foi gerado um cenário do tipo *Manhattan Grid* 10×10 sob uma região de 5,0625 km^2. Cada segmento de via possui 250 metros de extensão e duas faixas em cada sentido. O limite de velocidade dos veículos é de aproximadamente 50 km/h. Esse tipo de mapa é comumente utilizado para avaliar protocolos de redes veiculares em ambientes urbanos [de Sousa and Soares 2015] [Milojevic and Rakocevic 2014] [Araújo et al. 2014] [Garip et al. 2015].

Para criar a demanda de tráfego necessária no mapa, foram geradas 3000 rotas aleatórias de tamanho mínimo de 1 km. As rotas foram criadas escolhendo pares aleatórios de vias origem e destino. Para cada par, foi escolhido como rota a sequência de vias com o menor caminho que ligam esse par.

A duração de cada simulação foi de 120 segundos. Foram implementadas instâncias deste cenário com quatro demandas de tráfego diferentes. Nessas instâncias, a quantidade de veículos foi mantida constante em 50, 100, 150 e 200, respectivamente. Cada instância foi replicada 33 vezes. Todos os gráficos possuem intervalos com nível de confiança de 95%.

A potência de transmissão dos veículos foi de 7.1 miliwatts (mW) para a comunicação V2V, o que se traduz em um raio de comunicação de aproximadamente 300 metros. Já a comunicação LTE foi configurada de forma que um veículo tenha alcance da torre de transmissão a partir de qualquer posição do mapa. A Tabela 3 sumariza os principais parâmetros adotados nas simulações.

Parâmetro	Valor
Duração	120 segundos
Demanda de tráfego	50, 100, 150 e 200 veículos simultâneos
Potência de transmissão V2V	7,1 miliwatts (300 metros de alcance)
Frequência de transmissão de *beacons* para o controlador SDVN	1 Hz
Velocidade máxima dos veículos	50 km/h
Tamanho do cabeçalho dos pacotes	17 *bytes*
Tamanho dos dados de um *beacon*	17 *bytes*

Tabela 3. Parâmetros utilizados nas simulações do cenário *Manhattan Grid*.

Durante todas as simulações, um veículo foi escolhido para transmitir uma mensagem a cada 5 segundos. A região de destino da mensagem é uma região fixa com raio igual a 1500 metros. Toda a rota do veículo transmissor está contida nesta região. A Tabela 4 apresenta os valores das variáveis do protocolo *geocast* proposto utilizados nas simulações.

Parâmetro	Valor
Flow-Entry_Lifetime	120 segundos
Max_Waiting_Time	0.1 segundos
Max_Tries	3
Table-miss_Waiting_Time	0.1 segundos
Table-miss_Max_Waiting_Tries	3
Table-miss_Max_Tries	3
Vehicle_Entry_Duration	5 segundos
Guard_Range	20 metros

Tabela 4. Parâmetros utilizados das variáveis protocolo *geocast* proposto nas simulações.

4.2. Taxa de Entrega de Mensagens

A Figura 2 apresenta a taxa de entrega de mensagens. A taxa de entrega de mensagens refere-se à porcentagem de veículos presentes na região de interesse no momento do envio da mensagem que a receberam.

Observa-se que em todos os cenários avaliados o protocolo proposto apresenta uma taxa de entrega próxima a 100%. Isso ocorre devido às transmissões realizadas pelo controlador nos casos em que não é possível realizá-la utilizando a comunicação V2V. As perdas de pacotes, que ocorrem sobretudo em consequência das colisões, fazem com que a taxa de entrega não seja igual a 100%. Já o algoritmo *Flooding* apresenta uma taxa de entrega baixa nos cenários com baixa densidade de veículos e uma taxa alta nos cenários com maior densidade de veículos. Este fenômeno ocorre porque em cenários com poucos veículos, a rede formada pela comunicação V2V não é conectada, já em cenários com muitos veículos é formada uma rede conectada.

4.3. *Overhead* de Sinalização

A Figura 3 apresenta a quantidade média de pacotes enviados pelos veículos utilizando a comunicação V2V em cada um dos cenários avaliados.

Verifica-se que o algoritmo *Flooding* apresenta um crescimento linear acentuado no *overhead* de sinalização com o aumento da demanda de tráfego. Este comportamento acontece porque no algoritmo *Flooding* todos os veículos retransmitem a mensagem uma

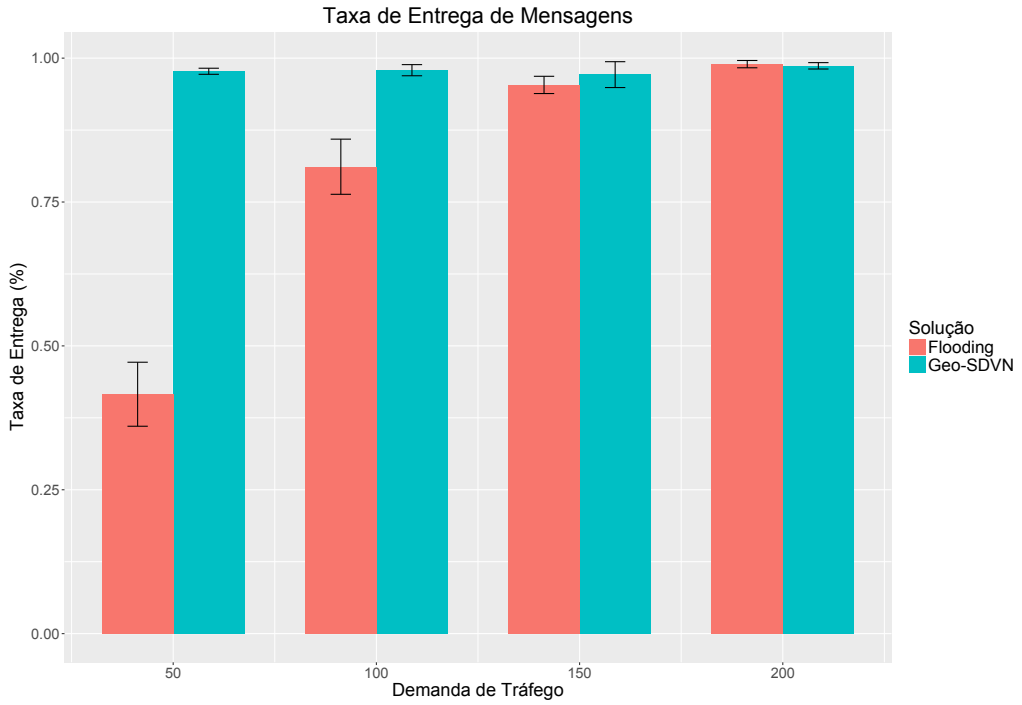


Figura 2. Taxa de entrega de mensagens.

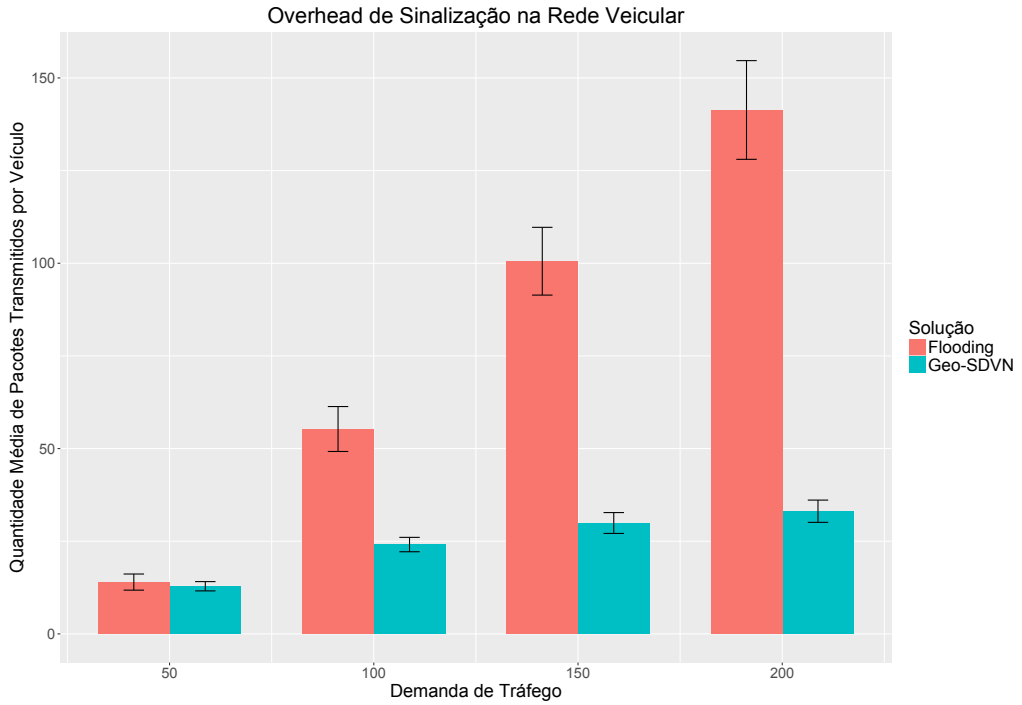


Figura 3. *Overhead* de sinalização.

vez, criando mensagens redundantes, o que resulta em uma tempestade de *broadcast*. Por outro lado, o protocolo proposto apresenta um baixo *overhead* de sinalização em todos os cenários. O principal motivo para este comportamento é o fato de apenas um subconjunto dos veículos da região de interesse serem escolhidos como nós retransmissores. Tal subconjunto é escolhido pelo controlador utilizado a heurística para calcular o conjunto dominante conectado "mínimo" de forma aproximada.

4.4. Atraso na Entrega de Mesagens

Por fim, a Figura 4 exibe o atraso médio na entrega das mensagens para os veículos da região de interesse.

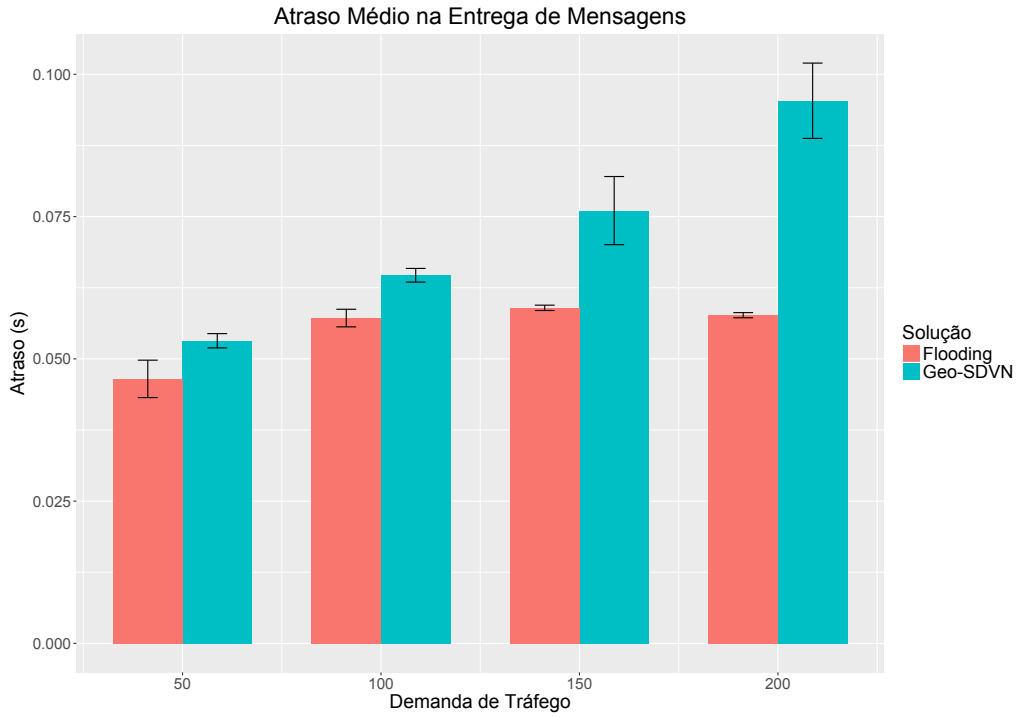


Figura 4. Atraso na entrega de mensagens.

Observa-se que, em todos os cenários, o protocolo proposto apresenta um atraso superior ao algoritmo de *Flooding*. Dois fatores contribuem para este fenômeno. Primeiro, a comunicação LTE possui um atraso maior que o da comunicação V2V [XiaoFang et al.]. Segundo, os mecanismos de recuperação utilizados pelo protocolo proposto quando são detectadas perdas de pacotes ocasionam em um aumento no atraso da entrega de mensagens.

5. Conclusões

Neste trabalho foi proposto um protocolo *geocast* para Redes Veiculares Definidas por *Software*. A arquitetura utilizada pelo protocolo dispensa a existencia de unidades de acostamento, fazendo o uso apenas da comunicação V2V e da comunicação entre veículo e controlador por meio da rede LTE.

Foram avaliadas a taxa de entrega, o *overhead* de sinalização e o atraso médio na entrega de mensagens. Os resultados foram comparados com um algoritmo de *Flooding*.

Verificou-se que o protocolo proposto possui uma maior taxa de entrega de mensagens e um menor *overhead* de sinalização V2V nos cenários avaliados. O atraso na entrega foi ligeiramente superior ao do algoritmo *Flooding*, principalmente devido à comunicação LTE.

Como trabalhos futuros, pretende-se analisar o desempenho do protocolo em outros cenários e analisar o *overhead* de sinalização do controlador SDVN. Será ainda verificada a melhor configuração de valores para as variáveis do protocolo. Também pretende-se comparar o protocolo com outros protocolos *geocast* propostos para redes veiculares tradicionais.

Referências

Araújo, G. B., Tostes, A. I. J., de L. P. Duarte-Figueiredo, F., and Loureiro, A. A. F. (2014). Um protocolo de identificação e minimização de congestionamentos de tráfego para redes veiculares. In *XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 207 – 220, Florianópolis.

Bachir, A. and Benslimane, A. (2003). A multicast protocol in ad hoc networks inter-vehicle geocast. In *The 57th IEEE Semiannual Vehicular Technology Conference, 2003. VTC 2003-Spring.*, volume 4, pages 2456–2460 vol.4.

Butenko, S., Cheng, X., Oliveira, C. A., and Pardalos, P. M. (2004). *A New Heuristic for the Minimum Connected Dominating Set Problem on Ad Hoc Wireless Networks*, pages 61–73. Springer US, Boston, MA.

Clark, B. N., Colbourn, C. J., and Johnson, D. S. (1990). Unit disk graphs. *Discrete Mathematics*, 86(1–3):165 – 177.

de Sousa, R. S. and Soares, A. C. B. (2015). Estimativa e sinalização de congestionamentos de tráfego através de redes veiculares V2V. In *XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, Vitória.

dos S. Alves, R., do V. Campell, I., and de S. Couto, R. (2009). Redes veiculares: Princípios, aplicações e desafios. In *Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC 2009*, chapter 5, pages 199–254. Sociedade Brasileira de Computação.

Garip, M. T., Gursoy, M. E., Reiher, P., and Gerla, M. (2015). Scalable reactive vehicle-to-vehicle congestion avoidance mechanism. In *12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 943–948.

Guha, S. and Khuller, S. (1998). Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387.

Hagenauer, F., Dressler, F., and Sommer, C. (2014). Poster: A simulator for heterogeneous vehicular networks. In *2014 IEEE Vehicular Networking Conference (VNC)*, pages 185–186.

He, Z., Cao, J., and Liu, X. (2016). Sdvn: enabling rapid network innovation for heterogeneous vehicular communication. *IEEE Network*, 30(4):10–15.

Joshi, H. P., Sichitiu, M. L., and Kihl, M. (2007). Distributed robust geocast multicast routing for inter-vehicle communication. In *Proceedings of WEIRD Workshop on WiMax, Wireless and Mobility*, pages 9–21.

Kazmi, A., Khan, M. A., and Akram, M. U. (2016). Devanet: Decentralized software-defined vanet architecture. In *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, pages 42–47.

Krajzewicz, D., Erdmann, J., Behrisch, M., and Bieker, L. (2012). Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138.

Ku, I., Lu, Y., Gerla, M., Gomes, R. L., Ongaro, F., and Cerqueira, E. (2014). Towards software-defined vanet: Architecture and services. In *2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, pages 103–110.

Kumar, A., Kumar, S., and Kumar, V. (2016). A novel energy efficient geocast routing algorithm for mobile ad hoc networks. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 2926–2929.

Liu, Y. C., Chen, C., and Chakraborty, S. (2015). A software defined network architecture for geobroadcast in vanets. In *2015 IEEE International Conference on Communications (ICC)*, pages 6559–6564.

Maihofer, C. (2004). A survey of geocast routing protocols. *IEEE Communications Surveys Tutorials*, 6(2):32–42.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Milojevic, M. and Rakocevic, V. (2014). Distributed road traffic congestion quantification using cooperative vanets. In *13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, pages 203–210.

Omar, H. A., Zhuang, W., and Li, L. (2013). Vemac: A tdma-based mac protocol for reliable broadcast in vanets. *IEEE Transactions on Mobile Computing*, 12(9):1724–1736.

OPEN NETWORK FOUNDATION (2013). Software-defined network (sdn) definition - open network foundation. `https://www.opennetworking.org/sdn-resources/sdn-definition`. acesso em: 26 de outubro de 2016.

Rahbar, H., Naik, K., and Nayak, A. (2010). Dtsg: Dynamic time-stable geocast routing in vehicular ad hoc networks. In *2010 The 9th IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pages 1–7.

Varga, A. and Hornig, R. (2008). An overview of the omnet++ simulation environment. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08, pages 60:1–60:10, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Voicu, R. C., Abbasi, H. I., Fang, H., Kihei, B., Copeland, J. A., and Chang, Y. (2014). Fast and reliable broadcasting in vanets using snr with ack decoupling. In *2014 IEEE International Conference on Communications (ICC)*, pages 574–579.

XiaoFang, L., Chi-Yuan, C., and Rouzbeh, R. Simulation comparison between lte and wi-fi in networks.

Zhang, H., Wang, R., and Larsson, T. (2014). Simulation of region-based geocast routing protocols. In *2014 International Conference on Connected Vehicles and Expo (ICCVE)*, pages 723–730.

Protection against attack D.o.S. in CAN and CAN-FD vehicle networks

Luiz Quintino, Alexei Machado

Electrical Engineering – Pontifícia Universidade Católica de Minas Gerais (PUC-MG)
Campus Coração Eucarístico
30.535-901 – Belo Horizonte – MG – Brasil

luiz.quintino@gmil.com, alexeimcmachado@gmail.com

Abstract. *For some time the vehicles adopted distributed electronic systems as a way to bring more comfort, active and passive safety to the occupants. With the increase of technological demand and innovations in connectivity, vehicles became attractive to hackers. Regardless of the goals of hacking, the future of automobiles goes through a wide range of related work evaluating the impacts of malicious people that take control of the vehicle's systems. In this article we evaluate the vulnerabilities of a vehicular data bus, CAN and CAN-FD. It is possible for an intruder to interfere with instrument panel display systems by sending forged messages to the driver and even compromise systems related to safety such as steering and brakes. Among the various types of possible attacks on the CAN bus, the D.o.S. (Denial of service) lacks of a deeper discussion in the literature. This article proposes a mechanism to protect the bus from this type of attack. A filter is designed to avoid that the bus be overloaded with false requests, providing access to authentic message exchanging.*

1. Introduction

The growth of automation in the vehicles and the connectivity with external devices allows modules to make decisions and take control of vehicular systems that controls comfort and safety (Larson, et al., 2008). Some systems control vehicle safety functions autonomously, without the driver intervention. Emergence Brake automatically reduces the vehicle's velocity in the presence of obstacles or pedestrians; Stop & Start shuts off the engine when the vehicle stops and restarts it automatically. These are some examples of features in modern vehicles related to the safety and comfort of the users. However, several studies have demonstrated weaknesses in the safety of electronic modules and current communication networks in vehicles (Larson, et al., 2008) (Wampler, et al., 2009) (Kleberger, et al., 2011).

Vehicles vulnerability becomes a point of attention and the hacker's interest on this type of technology is increasing. In a presentation on vehicle safety, Black Hat 2015, a hacker demonstrated vulnerabilities in a vehicle, in real time, remotely accessing their systems and gaining control over some modules. The involved vehicle manufacturer corrected the vulnerabilities, generating a recall of 1.4 million vehicles in the USA (Miller, et al., 2015). More recently Tesla and VW vehicles had a similar

problem and again, the modules needed to be re-flashed to protect the vehicle against malicious attack.

There is several kinds of attack that could be performed in a vehicular network. In this work we propose to minimize the effect of Denied of Service (D.o.S.) attack on any type of network that uses Controller Area Network (CAN) and CAN-Flexible Data-rate (CAN-FD) buses. The mechanism uses the arbitration layer of the transceiver to control the messages, avoiding the bus overload that occurs during a D.o.S. attack. Several tests were simulated on a bus mounted on a test bench to evaluate different conditions of attack. The goal is to have a minimum loss of authentic messages on the bus in the face of a D.o.S. attack in order to guarantee the correct operation of critical and security-related systems.

2. Vehicle network protection

Network protection has been studied extensively for communication between computers. The need for study came with the spread of Internet use to a growing number of purposes. Vehicle networks have more recent studies, mainly because there was no interest in attacking this kind of system. With the increase of use, the network communication in vehicles and the recent possibility of connecting the vehicle with the outside world, using cellphones connectivity, the vehicle became attractive and safety studies for this type of network tends to increase.

2.1 Types of vehicular network

The networks used in vehicles are presented in Figure 1. CAN is even the most used. Local Interconnected Network (LIN) is used for low cost systems complementing the CAN buses, used for sensors and no complex modules, Most and FlexRay are used for critical systems in luxury vehicles. There are studies to use Ethernet in the near future (Yong Kim, 2011).

Bus network	Characteristics	Bus type	Speed (Kbps)
LIN	Universal Asynchronous Receiver / Transmitter (UART)	Star; one wire	20
CAN	Carrier Sense Multiple Access (CSMA/CR)	Ring or star; twisted pair	1.000
FlexRay	Time Division Multiple Access (TDMA)	Star; Twisted pair	10.000
MOST	TDMA synchronous	Ring; Twisted pair or optical fiber	50.000 (optical fiber) 150.000
Ethernet	Commutation Full-Duplex	Star; Twisted pair or dual Twisted pair	100.000

Figure 1 - Characteristics of busbar used in vehicles (Yong Kim, 2011)

In the example of Figure 2, a typical vehicular architecture is presented with two independent CAN buses (CAN-1 and CAN-2). The buses are interconnected through a Bridge or Gateway. In this example one of the Electronic Control Unit (ECU5) has several external communication interfaces (internet, bluetooth, WiFi, etc.), a physical

connection to the bus, anda connector called On Board Diagnosis II (OBD-II) used to diagnose communication with external equipment connected to the bus.

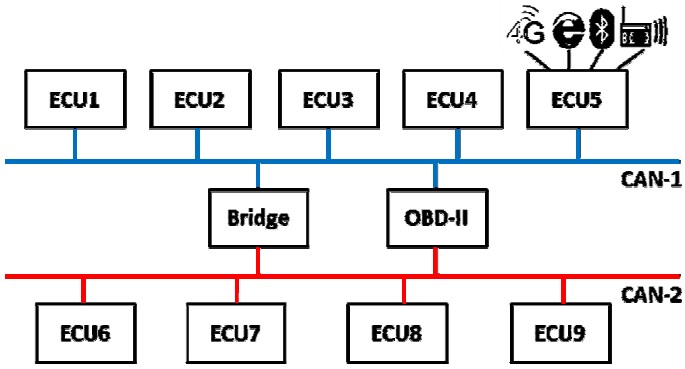


Figure 2 - Example of vehicular architecture (Wang, et al., 2014)

2.2 Characteristics of a CAN Network

The CAN network is based on the OSI model and has a layered structure where the lower layer is responsible for accessing the bus. One of the main features of the CAN network is to avoid collision, that means, there is no collision of messages on the physical bus.

A CAN message (Figure 3) is composed of an identifier (ID), control bits, 8 bytes of data, a Cyclic Redundancy Check (CRC), which is a mathematical calculation to test the integrity of a message, a bit of Acknowledgment (ACK) that is nothing more than a confirmation from the other nodes on the reception of a transmitted message, and a few bits reserved for error control in the End Of Frame (EOF) (ISO, 1999).

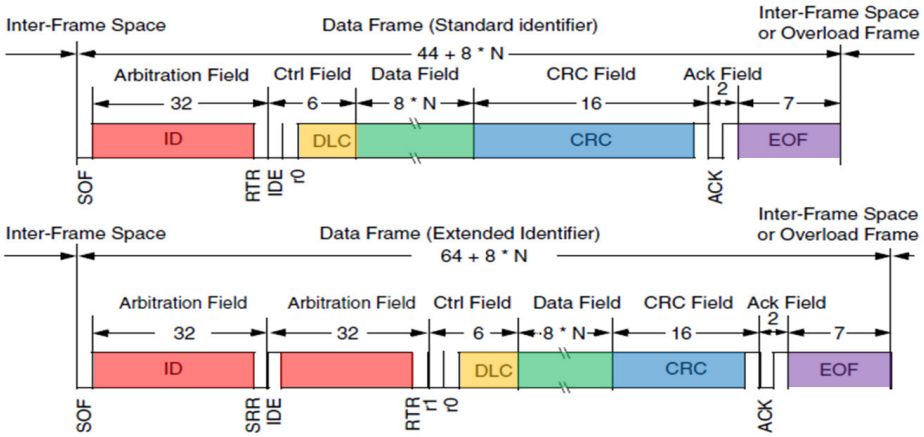


Figure 3 - Date frame of a message CAN (ISO, 1999)

The purpose of the identifier is to identify the message on the network. In this way, each different message has a unique ID that determines what that message is. As in the CAN network there is no information about the origin and destination of the message, it is the ID that will orienting the node that receives the message to filter and decide if each message is useful or not.

In addition to identifying the message, the ID also performs an important function of determining the priority of access to the CAN bus. The lower the ID number of a message, the higher its priority. In the case that two or more nodes want to transmit at the same time on the bus the ID is the first information transmitted to the bus. Each time a bit of the ID is transmitted this bit is read back by the sender node. Since bit 0 is dominant and bit 1 is recessive, smaller IDs will be dominant on the bus. When a node is transmitting but receives a dominant bit, the transmission is ceased, allowing the higher priority message to be transmitted. The node or nodes that fail to transmit will wait until the end of the next EOF, at which point the waiting nodes gain access to the bus again. Figure 4 shows a diagram representing two nodes, A and B trying to transmit at the same time. It can be observed that node A obtained access to the bus every time it wrote messages with the lowest ID value.

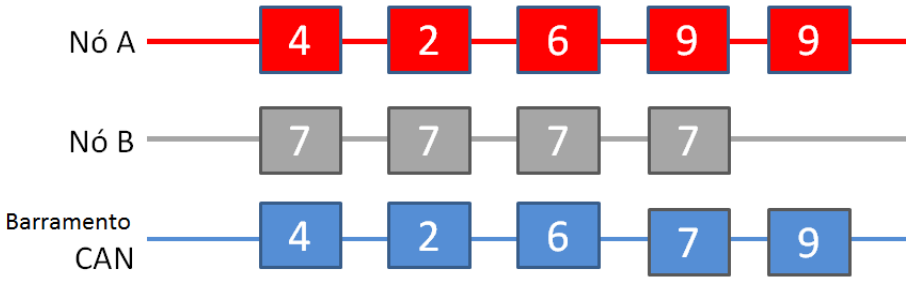


Figure 4 - Access arbitration to the CAN bus, adapted from Corrigan (Corrigan, 2008)

A CAN message may have IDs of 11 or 29 bits in length. The same bus may contain nodes that work with IDs of different sizes, as long as the node that will receive the message be able to identify it. A node that can only read 11 bit IDs may not receive a 29 bit message, the Identification Extended (IDE) bit is used to identify when a 29 bit ID is being used in the message.

The CAN bus has to work with a busload or “bus utilization rate” bellow 60%, to guarantee gaps in the bus so that all the modules can access it (Natale, 2008). In the development of the network busload, it must be calculated to evaluate the need to add new buses and to accommodate the amount of messages.

The CAN-FD uses the same principles of conventional CAN, but with greater speed and number of bytes. While the conventional CAN network can transmit at most 8 bytes of data at 1Mbps, the CAN-FD can transmit up to 64 bytes of data up to 12Mbps. For achieving a compatibility between CAN and CAN-FD, the same functional concept used in the CAN network was adopted in CAN-FD, and the high speed only happens in the data field phase, hence the name FD which means Flexible Data-rate, that is, it is possible to modify the transmission rate flexibly in the middle of the transmission.

2.3 Attacks and protection of vehicular networks

Nilson (Nilsson, et al., 1996) uses some criteria to classify and measure attack risks that make it possible to modify the firmware of electronic modules. Classification is based on Safety Integrity Level (SIL). Safety integrity is the likelihood of a safety-related system performing satisfactorily the required safety functions under all conditions in a given period of time. The Safety effect Level (SEL) classifies the effects of a malfunction on systems in levels 1 to 4 (Figure 5 - SEL according to the effect of the security threat).

Safety effect	Safety effect level (SEL)
Disastrous	4
Severe	3
Mediocre	2
Distracting	1

Figure 5 - SEL according to the effect of the security threat (Nilsson, et al., 1996)

In the work of Wolf (Wolf et al., 2004) and Nilson (Nilsson, et al., 2008) the weaknesses of the current networks are highlighted, since studies show protection in enough vehicular networks to allow malicious attacks and the possibility of success in creating such attacks.

Mahmud (Mahmud et al., 2006) states that a series of analysis should be carried out to study and balance the risks and costs involved in implementing a complete security policy, such as: classifying vehicle systems for risks inherent in its functionality as loss of comfort, loss of performance, accident risk, classifying critical modules, analyzing messages and signals that must be protected, analyzing impacts related to security breach in each class, and studying contingency plans.

Onishi (Onishi, 2012) estimates based on data from the US and Canada that damage from cyberattack has a potential loss of $ 56 million per year for every 1% of vehicles affected, considering the sales of a vehicle model. Onishi (Onishi, 2014) points out that vehicle system vulnerabilities are more complicated with respect to cyber-attacks than when compared to computers and the internet. Personal computers have about 2,000 components while vehicles have more than 20,000 components and up to 100 electronic modules with hundreds of megabytes of firmware code. Smit (Smith, 2016) presents techniques for gaining access to a vehicle's modules and networks, from hardware recognition, networking, firmware, electronics, and even to circumventing systems, entering commands, and modifying firmware's.

3 D.o.S. attack

The D.o.S. attack consists of flooding a network with messages in order to prevent its normal traffic (Nilsson, et al., 2008). It is important to know the D.o.S. concept in the computers to understand the mechanism used in the vehicle network. This prior knowledge is important, even knowing in advance that there is a difference in the network architecture and Internet protocol when compared to the CAN network. Likewise, solutions presented for one type of network do not apply directly to the other.

It is verified that there is no effective solution that guarantees 100% of protection for an Internet network against D.o.S. attacks and no solution has been found in the literature to protect a CAN network against a D.o.S. attack.

The consequences of a DoS attack on a vehicle are as devastating as the level of technology available in the vehicle. Decentralization of functions requires that systems communicate to make decisions. This means that a D.o.S. attack on a CAN network will completely stop any type of communication and put in risk the systems decisions. The more distributed the systems is and dependent on sensors and information that travel through the CAN bus, the bigger will be the consequences.

3.1 D.o.S. attack simulationon CAN network

A simulation environment composed of an architecture formed by several interconnected nodes was implemented for the experiments. The bus was configured to operate at 125kbps. The CANoe tool was used to create a message map with dozens of messages of different sizes and ID's. A message traffic flow was created in the message map between the nodes, with the definition of the nodes that will transmit each message and the periodicity thereof. The CANoe is a Vector tool designed to simulate vehicle buses. Vector is the biggest supplier of tools to monitor, controls, diagnostic and simulate vehicle buses. CANoe is a professional tool used from component developers and could simulate all kinds of vehicle busses other than CAN.

Figure 6a shows an image from an oscilloscope presenting a large empty space between messages on the bus. At this point the busload is about 14%. It is recommended that a CAN messaging architecture be designed to work with up to 60% busload, ensuring enough spaces for all packages to be delivered with minimal delay (Corrigan, 2008). The arbitration mechanism of the CAN network guarantees that messages with a smaller ID will have priority access and will guarantee less delays in relation to less priority messages. When a CAN network is under attack there is no space between messages (Figure 6b). The attacker uses the lowest ID preventing authentic messages from being transmitted.

Figure 6 – a) Normal CAN bus 14% bus load; b) CAN bus under attack 100% bus load
Source: from author

3.2 Attack protection for D.o.S. attack over a CAN and CAN-FD network

A way to mitigate D.o.S. attacks on a CAN bus could be to limit the number of messages activing the bus from the same node in a period of time. As this type of control is not possible in a centralized way, an alternative was designed for each node to have autonomy to manage a control, filtering the message before sending it.

Since each node is responsible for managing the access to the bus and each node needs a transceiver, the method presented in this paper uses the CAN transceiver to control the message flow during an attack. The idea is based on a constant flow control using a hardware filter implemented in the transceiver with the capability to limit the access to the bus. Since every node has a transceiver, it is guaranteed that access to the bus will be controlled. The filter is able to limit the number of messages that a node will transmit within a period of time, allowing the other nodes to also have access to the bus.

Each time a message arrives from the micro-controller to be sent, the transceiver must check the value of an internal counter. If the counter is less than a value set as a configurable filter parameter the message is transmitted and the counter is increased. If the counter is greater than the set value, the message will not be sent and the node will wait until the counter value is decremented, allowing a limited number of consecutive messages to be sent. The way to decrement the counter is related to a timer configured to count a time t, which is also a configurable parameter. Each time the timer reaches the set time it will decrement the counter until it reaches zero, so the timer will generate a delay in the transmission of the message during an attack.

This mechanism allows messages that are transmitted within normal communication intervals, which generate busloads below 60%, do not have their transmission altered by the filter, and in this way there will be no packet loss or transmission delays.

In case of a D.o.S. attack, the node promoting the attack will have its messages controlled by the filter and after sending a sequence of consecutive messages within the established time limit, it must wait for a delay before being able to access it again. This waiting time allows the other nodes to compete for access to the bus and thus the main functions of the vehicle will be performed even during an attack.

The filter algorithm could be incorporated into both a CAN transceiver and a CAN-FD. The implementation can be done by software but it would be more indicated by hardware, for hindering the possibility of adulteration of the configurations.

3.3 Filter parameterization

The configuration values for the number of consecutive messages sent and time to decrement the counter was evaluated in order to find an ideal value for maximum efficiency and minimum loss of packages. This generates a delay in the next sending of messages by the node. In this way it would be possible for the transceiver to be programmed from the factory with a default setting value that cannot be changed. The measurements in the simulator have the objective of finding the optimal values for the parameters that represent the minimum of packet loss with the minimum influence on the normal communication of the bus.

During the tests, two types of information were collected: the busload of the network and loss of message packs. The busload rate is important to analyze the efficiency of the filter in keeping the bus in operational condition, seeking not to change the flow of messages compared to normal operation. Similarly, busload information will also help to analyze network conditions during an attack and the influence of the filter in attempting to establish a control in the message flow.

4. Experiments

Initially the CAN network was configured to 125kps, with a busload of 10%. The number of messages or packets transmitted under normal conditions was measured. Five measurements were taken to register the average of authentic messages transmitted within 1 second. It was observed that some messages have a higher periodicity than others, and therefore are transmitted more often.

An attack was introduced using the 0x90 ID, as being an ID smaller than the lowest ID used in the messages that were originally being transmitted (Figure 7). The time between the attack messages was been set in the 960µs range, since a CAN message at a rate of 125kbps has this duration time with 8 byte of Data Length Control (DLC). This attack generated a 100% busload and a loss of 100% of the authentic messages, completely disabling the bus. In this way, no node could communicate with others and services that depended on messages, commands or information from other nodes would be unavailable. It was also noted that the number of packets or messages sent was about 993 messages, rather than 121 in normal operation, which means the maximum number of queued messages that can travel on the bus in the period of 1 second.

To demonstrate the influence of the periodicity on the message identifier, an attack test was performed with the ID 0x1E40000 that is bigger than the authentic ID's used. Despite the busload having reached 100% saturation, there was no packet loss and all 121 authentic messages were successfully transmitted without delays, since all authentic IDs have priority over the attacker ID.

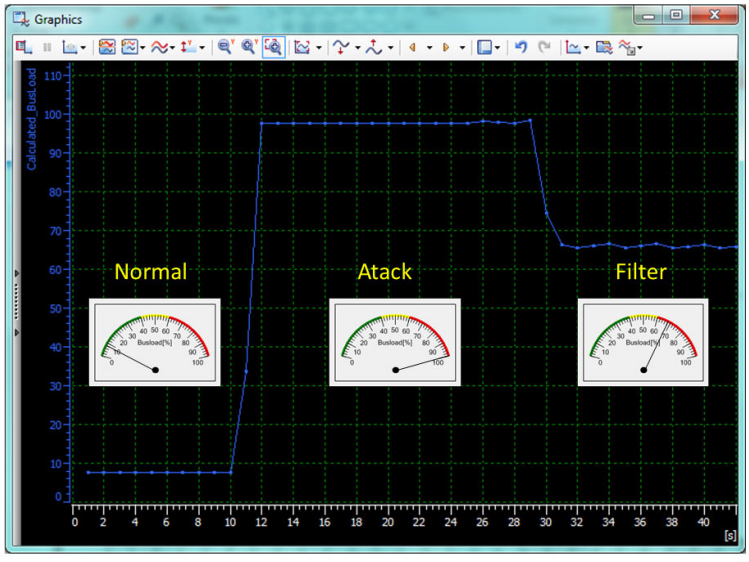
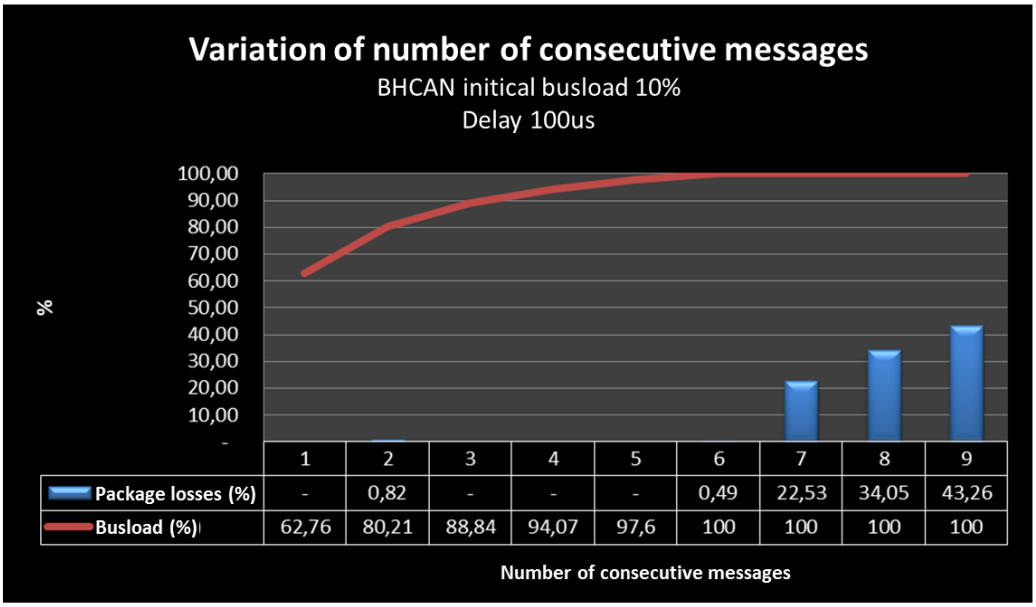


Figure 7 - Busload view during filter test
Source: from author

A test was also made considering the CAN bus with 125kbps and initial busload of 10%, with message delay of 100µs and varying the number of consecutive messages from 1 to 9. The result is shown in Figure 8. It can be observed that the number of consecutive messages has a direct influence on busload and packet loss during an attack. A higher number of consecutive messages that the node sends will generate a higher busload rate during an attack and the greater the number of packets lost.



	1	2	3	4	5	6	7	8	9
Package losses (%)	-	0,82	-	-	-	0,49	22,53	34,05	43,26
Busload (%)	62,76	80,21	88,84	94,07	97,6	100	100	100	100

Figure 8 - Filter test with variation of the number of consecutive message loss of packs (%) and busload rate (%) in function of the variation of the number of consecutive messages allowed by the filter varying from 1 to 9.

Source: from author

4.1 Message filter settings

In order to obtain an optimum calibration value of the filter parameters, several conditions were tested, varying the initial busload, waiting time, number of consecutive messages and bus speed. In total, 21 delay times, 9 consecutive message variations for 3 initial busload variations were made for CAN at 125kbps. These measurements were repeated for CAN at 500kbps with 13 time delay variations, 9 consecutive message variations for 2 initial busload variations, repeating the measurements 5 times to obtain the average of packages, making a total of 4005 measurements.

In order to better represent the values found in the tests, the graph of Figure 9 was plotted. The vertical axis shows the busload rate as a function of the delay time in the attack messages. Each curve represents the variation of the number of consecutive messages. The variation in delay time was started with a range of 100µs, representing 10% of the message size. As can be observed this variation only had an effect on the busload rate after a delay of 1000µs, which represents the 100% of message length. Due to this measured characteristic was used a new interval of 500µs in the delay time, that is, 50% of the message length.

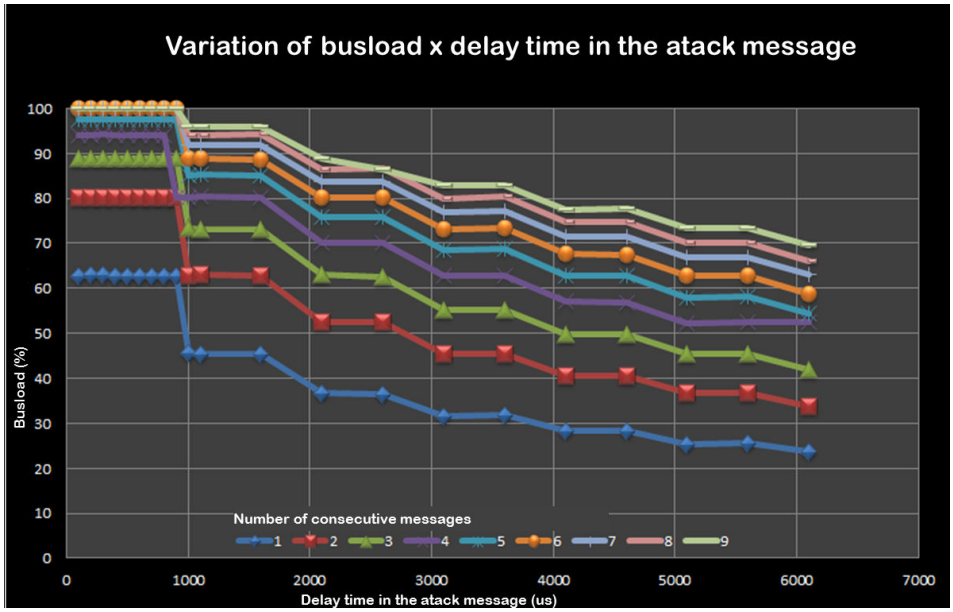


Figure 9 - Busload measurement during D.o.S. attack with the filter active
Source: from author

The tests were made for CAN with initial busload of 20% and 60% at 125kbps and at 500kbps. It can be observed (Figure 10) that the margin for the filter to act with the initial busload at 60% is very small, but it was still possible to find calibration values that kept the busload below 100% during an attack. In the same way, with 60% of initial saturation there were losses of packages that reached 100% for a certain range of calibration of the parameters, however, the filter was still effective in the green area highlighted in the graph, in which there was no loss of package (Figure 11).

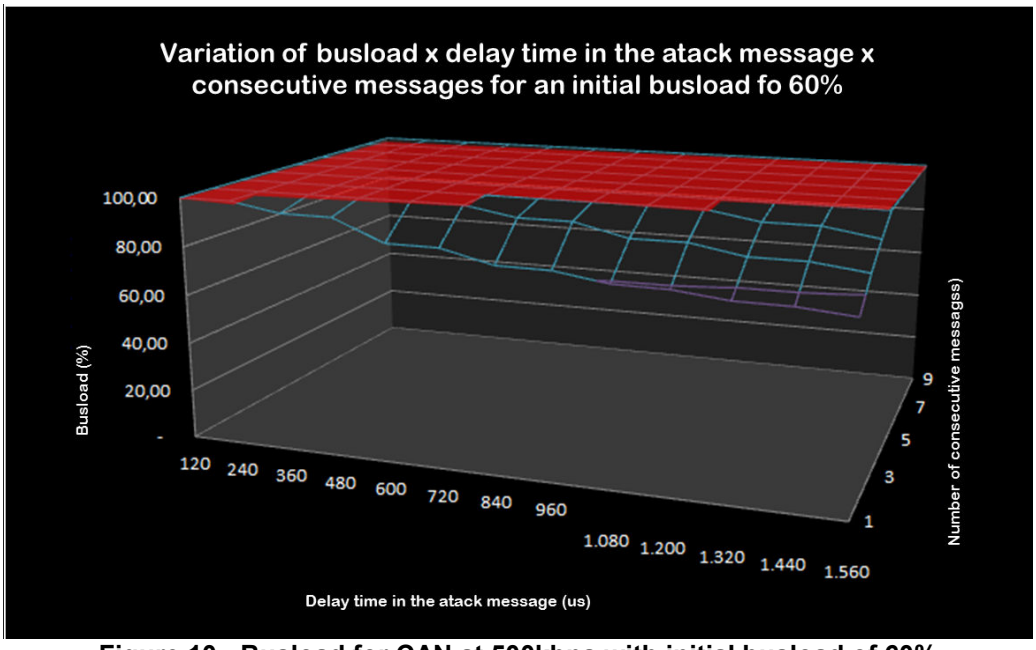


Figure 10 - Busload for CAN at 500kbps with initial busload of 60%
Source: from author

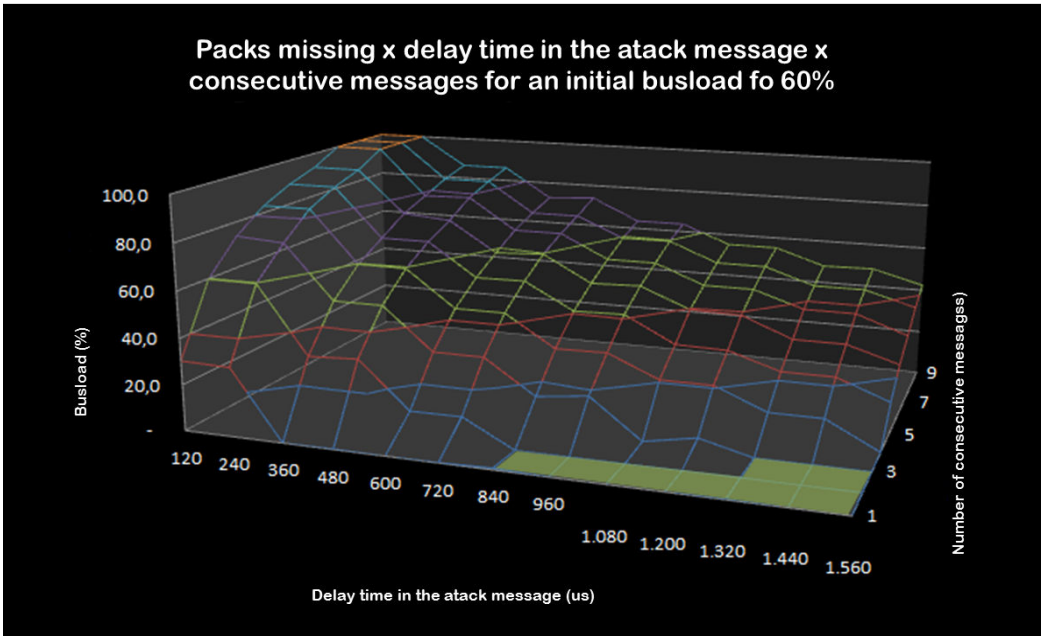


Figure 11 - Messages losses for CAN at 500kbps with initial busload of 60%
Source: from author

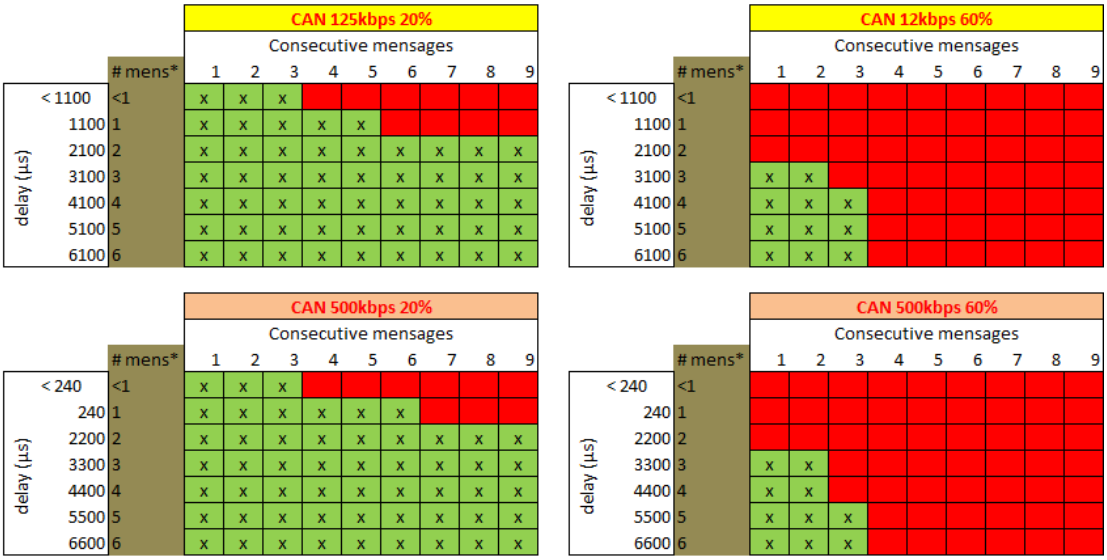
4.2 Analysis of results

The performed measurements indicate that the filter can be efficient even in the most critical condition of busload, 60%, and with a baud rate of 500kbps in all test conditions with the proper calibration adjustments. It was noted that the initial bus saturation and the baud rate significantly modified the effectiveness of the filter. Figure 12a shows all condition tested. The green area is where the filter was efficient in dealing with the D.o.S. attack. The Figure 12b summarizes the filter settings overlapping. It may be noted that there is a region where the filter was able to deal with the attack under all conditions of the bus. This means that the yellow region values in the frame can be used for all bus conditions tested. In the table, the delay time was indicated in relation to the size of a message, in order to refer to any bus speed. As was demonstrated, the size of a message on at 125kbps bus is 970µs and at 500kbps it is 240µs.

A single node can send several different messages, with different periodicities and IDs. Are the parameters found for the filter enough to guarantee that in normal condition the messages will not be delayed? There are two ways to analyze this question. The first one is regarding to regular messages; the second one, for diagnostic service messages.

For regular messages in a CAN bus with 125kbps, the modules use to have periodicity between 100ms and 2.000ms. In general one module sends from one to 5 different messages. Consider the message length equal to 1ms and the filter adjusted to 3 consecutive messages with a delay of time of 5ms on average. Considering the worst case within 5 messages transmitted in 100ms of periodicity. The total time to transmit 5 messages will be 3x1ms + 5x1ms + 2x1ms. This means that after three messages the module will wait 5ms to send the other 2 messages in a total time of 10ms. For 100ms of periodicity the module will send these messages again after elapsed 90ms. This

means that in this worse case, the delay time from the filter is not significant. To simplify, *total_time* = (**round**((*<number_of_messages>* + 1) / 3) * 5 + *<number_of_messages>*) * 1ms. Even if we consider the module sending double of regular messages using the maximum period, this means all messages will be sent in a total time of (**round**((10 + 1) / 3) * 5 + 10) * 1ms = 25ms << 100ms. In a real situation the vehicle uses to have few numbers of high periodic messages comparing with low periodic messages.



(*)mens = number equivalent to the messages lenght, deppending of the CAN bus speed.

Figure 12a – Filter configuration map
Source: from author

Figure 12b – Filter configuration map
Source: from author

Diagnostic messages can reach up to 255bytes and there are firmware upload services that have multiple kbytes of data. Transmitting this kind of messages the filter will generate a considerable over time in transmission. If we consider the effect of the filter we need add the time of 5 messages for every 3 messages really sent. A diagnostic message could send 7 bytes of data per each message, because of the Transport Protocol implemented over CAN, to transmit big data. For a 2MB file, for example, divided per 7 bytes we would have 285,715 diagnostic messages necessary. Appling the formula, the total time to this transmission with the filter working is 761.905ms or 12.32 minutes.

Without filter the total time is 4,76 minutes. This means 259% over the original time. If we consider that there are currently firmware upload processes that take 1h20m, the filter will increase the total time to 3h30m, which would be unacceptable.

The diagnostic messages always have a low priority ID, typically over 0x700 for 11 bit ID's. As has been shown previously diagnostic messages do not represent a risk for bus attack. A simple way to solve diagnostic message transmission is to bypass the diagnostic messages in the filter without compromise the protection against D.o.S. attack.

5. Conclusion

In this work, a method was presented to minimize D.o.S. attacks on any type of network that uses CAN and CAN-FD buses by creating a filter in the arbitration layer to measure the messages, avoiding bus overload and packet loss which occurs during a D.o.S. attack. The results of tests made in a simulator confronted with information from a real network were presented.

The goals of having, during a D.o.S. attack, a minimum loss of authentic communications and do not fail of critical or security-related systems, was overpassed and the proposed filter could be parametrized to avoid completely the losses of authentic messages at the buss. The filter is able to limit traffic overhead, allowing authentic messages to be transmitted between nodes even during an attack.

References

Corrigan, Steve. 2008. Introduction to the Controller Area Network - CAN. *Texas Instruments - Application Report.* July de 2008.

ISO. 1996. ISO 11989 - CAN Network. *CAN Network.* 1996.

Kleberger, Pierre, Olovsson, Tomas e Jonsson, Erland. 2011. Security aspects of the in-vehicle network in the connected car. *IEEE - Intelligent Vehicles Symposium IV*. 5-9 de Junho de 2011.

Larson, Ulf E., Nilsson, Dennis K. e Jonsson, Erland. 2008. An Approach to Specification-based Attack Detection for In-Vehicle Networks. *2008 IEEE Intelligent Vehicles Symposium.* 4-6 de Junnho de 2008.

Mahmud, Syed Masud e Shanker, Shobhit. 2006. In-Vehicle Wiereless Personal Area Network (SWPAN). *IEEE Transactions on vehicular technology.* maio de 2006, Vol. 55, 3.

Miller, Dr. Charlie e Valasek, Chris. 2015. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA.* Agosto de 2015.

Natale, Marco di. 2008. Understanding and using the Controller Area Network. 30 de October de 2008, p. 47.

Nilsson, D. K. e Larson, U. E. 2008. Conducting forensic investigations. *First International Conference on Forensic Applications and Tech-*. 21-23 de janeiro de 2008.

—. 2008. Simulated attacks on CAN buses:. *5th IASTED Asian Conference on Communication Systems and Networks.* 2-4 de Abril de 2008.

Nilsson, Dennis K., Phung, Phu H. e Larson, Ulf E. 1996. Vehicle ECU classification based on safety-security characteristics. *Chalmers University of Technology.* 1996.

Onishi, Hiro. 2014. Approaches for Vehicle Cyber Security. 2014.

—. 2012. Paradigm change of vehicle cyber security. 2012.

Smith, Craig. 2016. *The Car hacker's handbook - A guide for penetration tester.* 2016.

Wampler, David e Fu, Huirong. 2009. Security Threats and Countermeasures for Intra-Vehicle Networks. *IEEE Computer Society - Fifth Inernational Conference of Inforation Assurance and Security.* 2009.

Wolf, M., Weimerskirch, A. e Paar, C. 2004. Security in Automotive Bus. *Workshop on Embedded IT-Security in Cars.* 11-12 de novembro de 2004.

Yong Kim, Masa Nakamura. 2011. Automotive Ethernet Network Requirements. *IEEE 802.1 AVB Task Force Meeting.* Março de 2011.

Trilha Principal do SBRC 2017
Sessão Técnica 13
Redes Móveis e Dinâmicas

Measuring Burden and Routing Fairness in Pocket Switched Networks

Tekenate E. Amah[1], Maznah Kamat[1], Kamalrulnizam Abu Bakar[1], Waldir Moreira[2, 3], Antonio Oliveira-Jr[4], Marcos A. Batista[5]

[1]Department of Computer Science, Faculty of Computing, Universiti Teknologi Malaysia (UTM), 81310 Skudai, Johor, Malaysia

[2]Fraunhofer-AICOS - Porto, Portugal

[3]PPGMO, Federal University of Goiás - Catalão, Goiás, Brazil

[4]Institute of Informatics, Federal University of Goiás - Goiânia, Goiás, Brazil

[5]IBiotec, Federal University of Goiás, Catalão, Goiás, Brazil

amatek008@yahoo.com, {kmaznah,knizam}@utm.my, waldir.junior@fraunhofer.pt, antonio@inf.ufg.br, marcos.batista@pq.cnpq.br

Abstract. *A Pocket Switched Network (PSN) is formed by users carrying portable handheld devices such as smartphones and tablets, which store messages, carry them from one point to another via physical movement, and forwards them when a communication opportunity arises. The success of the network thereby depends on the willingness of users to participate. PSN protocols tend to subject most of the routing burden on only a smaller set of popular nodes. This results in drastic resource consumption on popular nodes, and may eventually lead to user dissatisfaction, withdrawal, and performance degradation of the network. The key to ensuring fairness in PSN routing lies in the ability to estimate the burden on nodes, utilize this knowledge to provide an acceptably fair utilization of node resources, and evaluate the level of fairness achieved. This paper is concerned with measuring: (i) the burden routing impacts on nodes; and (ii) the fairness of routing algorithms based on the distribution of this burden. First, we propose a Global Relative Burden Detection (GReBurD) mechanism to estimate the burden on nodes. Simulation experiments show that GReBurD is non-scenario specific and better infers the actual burden on nodes as compared with existing approaches. Next, we propose a new metric for evaluating the fairness of PSN forwarding algorithms that gives a better interpretation of the level of fairness implied.*

1. Introduction

Pocket Switched Networks (PSN) are based on the store-carry-forward (SCF) communication paradigm, in which messages are stored in node memory, physically carried from one point to another via user movement, and forwarded (through an available wireless communication interface such as Bluetooth or Wi-Fi) to another node when a communication opportunity arises. With respect to SCF-based networks, Mtibaa and Harras (2013) define fairness as the relative equality in the distribution of resource usage among neighbouring nodes in the network. Therefore, a forwarding algorithm is

absolutely fair if it subjects equal burden (i.e., resource expenditure) on all nodes. Absolute fairness leads to undesirable performance degradations in PSNs. This is due to the small world nature of the network (Orlinski and Filer, 2012), in which a relatively small set of "popular nodes" have more encounter opportunities than others (Tsvetovat and Kouznetsov, 2011; Muchnik *et al*., 2013), thereby causing large variances in forwarding ability. Such nodes are often carried by popular users (e.g., politicians) or by those who travel around more frequently than others (e.g., deliverymen).

Nevertheless, it is important to achieve an acceptable level of routing fairness in order to ensure collaboration in PSNs. Since user devices are fairly homogeneous (Pujol *et al*., 2009) in terms of resources such as battery capacity and computational capability[1], subjecting only few nodes to most of the burden in the network results in drastic resource consumption on them, and may eventually lead to user dissatisfaction, withdrawal, and performance degradation of the network (Amah *et al*., 2016). Unfortunately, most routing protocols operate in this manner because their forwarding techniques and utilities are biased towards popular nodes (Pujol *et al*., 2009). Considering that nodes are owned by humans, who may be unwilling to allocate most of their resources to routing, ensuring user participation is paramount. Hence, the success of routing solutions in real-world implementation lies in their ability to guarantee an acceptable level of fairness by sparingly utilizing the resources available on each node, instead of subjecting only few nodes to most of the burden in the network.

Besides the small world nature of PSNs, the distributed nature of the network makes it even more challenging to achieve fairness. Network designers need to be able to estimate the burden on nodes, utilize this knowledge to provide an acceptably fair utilization of node resources without significantly compromising network performance, and evaluate the level of fairness achieved. Existing approaches infer the burden on nodes from buffer information. However, inferring the burden on nodes from buffer information limits existing solutions from guaranteeing actual fairness in realistic scenarios, especially when other resource constraints (e.g., energy and processing) are considered. Focusing on the buffer alone may not achieve the desired level of fairness, because there is no guarantee that nodes will have equal buffer sizes. Furthermore, buffer occupancy is not a good indicator of how much burden a networking node is subjected to, due to the fact that node resources could be overwhelmed even without having a significant portion of the buffer occupied.

In this regard, our contribution is twofold. With respect to the first contribution (cf., Section 2), we analyse existing techniques for estimating the burden on nodes and identify their major drawbacks. We then propose the Global Relative Burden Detection (GReBurD) mechanism for estimating burden in PSNs without the identified drawbacks. Particularly, simulation experiments show that GReBurD is non-scenario specific and better infers the actual burden on a node, which is in accordance with energy consumption. With respect to the second contribution (cf., Section 3), we analyse existing metrics for measuring fairness and point out shortcomings of using these metrics to evaluate PSNs. We then propose a metric that is free of the identified shortcomings when used to measure routing fairness in PSNs. Our proposal retains all the properties of a desirable fairness metric, and allows for a better interpretation of the

[1] Note that allocated storage space may be according to user discretion, thus, may vary across nodes.

level of fairness implied. Finally, we present our conclusions and future work in Section 4.

2. Measuring Node Burden in PSNs

In this section, we analyse existing techniques for estimating the burden on nodes, identify major drawbacks in PSNs, propose an improved measure for burden, and support our claims through simulation experiments.

2.1. Existing measures for node burden

Authors often infer the burden on nodes from buffer information in order to achieve fair routing in PSNs. Pujol *et al*. (2009) forward messages to relay nodes based on the size of their message queue in order to balance load among nodes in the network. Grundy and Radenkovic's (2010) forwarding approach prioritizes nodes that have a higher percentage of remaining storage capacity, in order to distribute load away from popular nodes to less popular nodes. Mtibaa and Harras (2013) infer the burden on nodes from the number of messages they can carry, in order to ensure an efficiency fairness trade-off in forwarding. In order to achieve a tuneable trade-off between efficiency and fairness, Akestoridis *et al*.'s (2014) criterion for accepting a message is based on a utility derived from buffer occupancy.

Unfortunately, there is no guarantee that users in real-life will allocate fairly equal amount of storage space to routing. Therefore, comparing nodes by their buffer occupancy may be unfair in certain scenarios. For instance, depending on the amount of storage space allocated to routing, a node with 50% buffer occupancy may have received more messages and thereby expended more energy than another node with 75% buffer occupancy. By comparing buffer occupancies, messages continue to be directed towards the node with 50% buffer occupancy instead. This leaves a lot of room for error in real-world implementation, considering allocated storage resources may not be homogeneous after all.

This reveals that Akestoridis *et al*.'s (2014) approach will subject more burden on nodes that have assigned more storage space to routing since burden is considered as the ratio between remaining storage space and total storage capacity. Grundy and Radenkovic (2010) make forwarding decisions based on the percentage of available buffer space such that messages are directed towards nodes with less buffer occupancy. In a scenario where some nodes have allocated more storage space to routing than others, the former will be subjected to unfair treatment (e.g., more energy will be consumed from them), which cannot be detected by buffer occupancy.

Apart from the issue of different buffer sizes, buffer occupancy is still not a good indicator of how much burden a node is subjected to, since other resources could be overwhelmed without occupying a significant portion of the buffer. For instance, if the rate of receiving and sending messages is high, utilizing a fixed buffer occupancy threshold to ensure fairness may not be suitable. In that case, popular nodes may have used up all their energy allocated to routing without even reaching the threshold. Moreover, unlike the Internet, PSNs are constrained in terms of resources allocated to routing. Therefore, Internet-based approaches may not be suitable: even before the issue reflects on the buffer, energy usage may have exceeded the allocated quota. Unfortunately, most related work is oblivious of energy. Hence, the node continues to

function in the network even after allocated energy would have been exceeded, and the actual burden on popular nodes is not detected early enough.

Due to differences in the rate of sending and receiving messages, a higher buffer occupancy may not always mean higher burden, and vice versa. It is likely for popular nodes to have higher buffer occupancy most of the time, since they receive more messages. However, this may not always be the case, as popular nodes are also able to deliver more messages than less popular nodes due to higher encounter opportunities. As a result, depending on the rate of message generation, popular nodes may free their buffers faster than less popular nodes, especially when there is a drop in data traffic (e.g., after traffic bursts).

Detecting the burden on nodes via energy could also come to mind[2]. However, unless the portion of energy allocated to routing can be monitored, overall energy consumption (i.e., residual energy) itself is also not a good indicator of the burden on a node. The chances that a device is low on battery due to other applications besides routing cannot be ruled out in real-life. This implies that determining the burden routing has impacted upon a node from residual energy may not be fair. For instance, a user conserving his battery for later use may find that most of it has been utilized for routing instead, while less would have been utilized if he was less conserving. Hence, inferring fairness from buffer occupancy or residual energy is only suitable for evaluation in controlled environments such as experimental simulations. To prevent bias in performance evaluation, simulation environments can be controlled to assume that device resources are consumed through participation in the network alone. Likewise, equal buffer size could be assigned to every node. This is however, not true in real-life, and should not be used as a basis of designing burden measures intended for real-world implementation. To address this gap, we propose an effective mechanism for inferring the burden on PSN nodes in Section 2.2.

2.2. Proposed measure for node burden

The approach employed here, namely Global Relative Burden Detection (GReBurD), determines the burden on a node from transmitted (i.e., received or sent) messages. For each node a, GReBurD maintains a counter for transmissions (i.e., received messages, excluding messages for which a is the destination, plus sent messages, excluding messages for which a is the source), $T(a)$. At the end of every time slot, ΔTb_i ($i \in [1, \infty]$), the current burden on a, $B_i(a)$, is computed by Equation 1, where $B_{i-1}(a)$ represents the burden on a in the previous slot, ΔTb_{i-1}, if there was one (i.e., if $i > 1$). After computing the burden, $T(a)$ is reset to 0 for the next time slot, ΔTb_{i+1}. Without resetting the transmission counter, the burden on nodes does not change even after a long period of inactivity.

$$B_i(a) = \begin{cases} T(a), & i = 1 \\ \big(T(a) + B_{i-1}(a)\big)/2, & i > 1 \end{cases} \qquad Equation\ 1$$

The burden at a given instant may be used to make forwarding decisions. In that case, the instantaneous burden can also be computed with Equation 1 (note that the

[2] Note that energy-aware routing solutions consider energy level in order to improve routing performance, rather than (achieving fairness by) making routing decisions according to the burden routing itself has impacted on nodes.

counter is not reset until the current time slot is elapsed). Based on the proposed mechanism, the instantaneous burden on nodes increase as they continue to participate in forwarding. At the beginning of a new time slot, the instantaneous burden on nodes is reduced since the transmission counter is reset to 0. Hence, the choice of ΔTb_i reflects the restoration experienced during the rest periods, such as when a device could be recharged, as opposed to periods of high traffic such as rush hours. For this, we have selected a period of six hours and leave details of this to future work. This choice is based on results from several trials, where we found that it is likely to reset the transmission counters when the user is either at home or at work, i.e., when there are least transmission activities in the network.

2.3. Evaluation of proposed measure for node burden

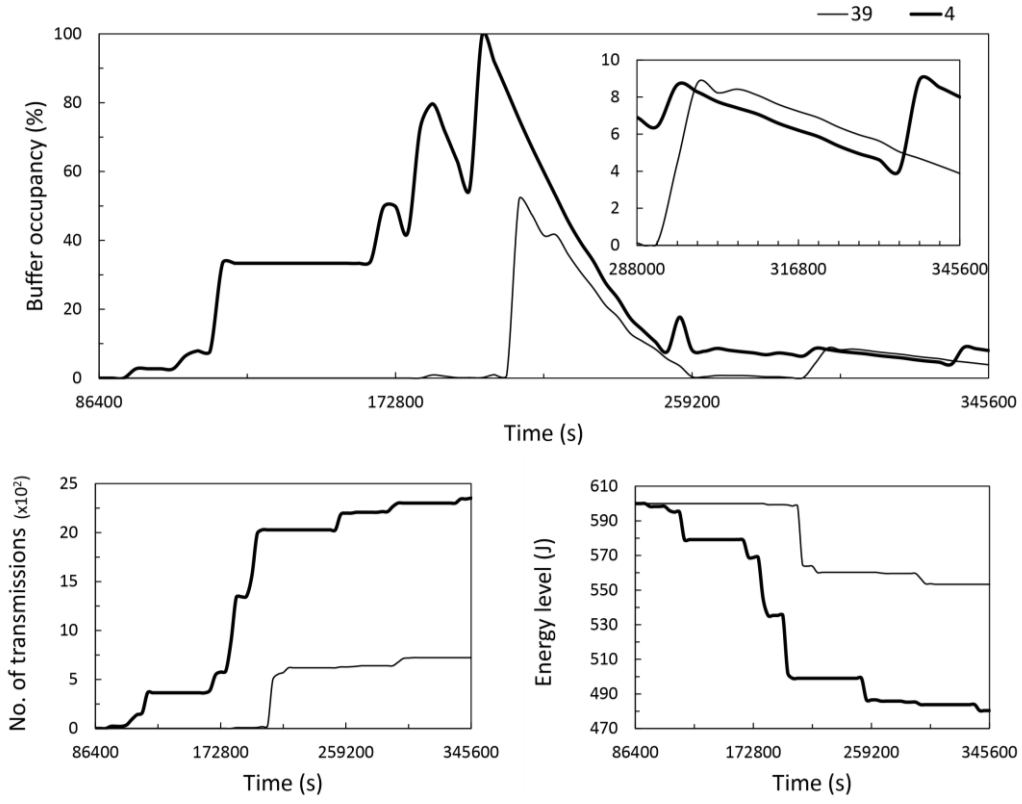
Our evaluation was carried out in the Opportunistic Network Environment (ONE) simulator (Keränen *et al*., 2009). We assume a scenario in which portable handheld devices, carried by mobile users in a city, convey data between static sources (e.g., sensors) and destinations (e.g., base stations) placed in strategic locations. Sources are situated around homes while destinations are located around bus stops and meeting spots in the Helsinki scenario in ONE simulator. Nodes forward messages according to the PRoPHET routing protocol (Grasic *et al*., 2011). For evaluation purposes, all nodes are given equal resources in terms of buffer size and initial energy. Since our interest is only on energy consumed due to message forwarding, we set scan energy (i.e., energy consumed from device discovery), scan response energy (i.e., energy consumed from device discovery response), and base energy (i.e., energy consumed in idle state) to 0, in order to avoid bias. All mobile nodes move according to the Working Day Movement model (Ekman *et al*., 2008), which presents realistic human movement patterns. The parameters used for the simulation setup are summarised in Table 1.

Table 1. Simulation parameters.

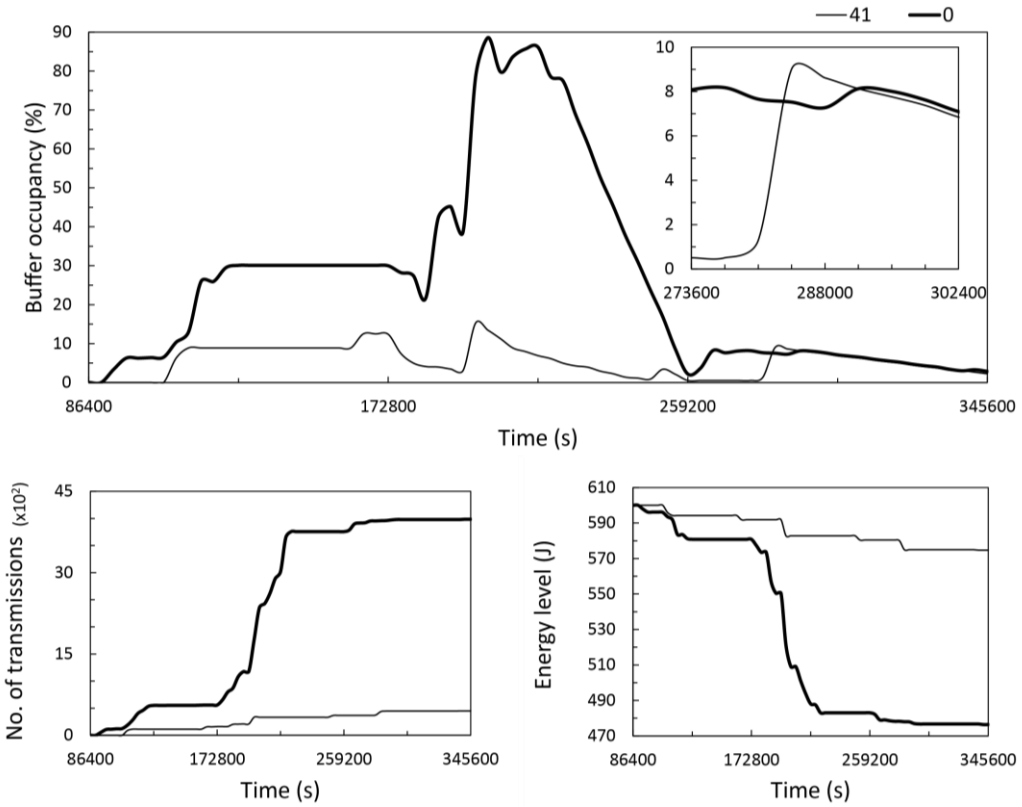
Parameter	Value	Parameter	Value
Wireless interface	Bluetooth	Receive/transmit energy	0.08 mW/s
Transmission rate	2 Mbps	No. of source nodes	80
Buffer size	10 Mb	No. of destination nodes	36
Message size (uniformly distributed)	10 Kb to 15 Kb	No. of buses	18
Simulation area	7 km × 8.5 km	No. of taxis	50
Initial energy	600 J	No. of pedestrians	416

We have chosen a simulation duration of 4 days. 1 day is allowed before message generation, for warm-up. The warm-up period allows nodes to acquire enough routing information to make forwarding decisions according to the routing protocol in use – in this case PRoPHET. Data is generated for three days: 1st day, high generation rate (1msg/hr per src); 2nd day and 3rd day, low generation rate (1msg/24hrs per src). This allows us to investigate the performance of the burden measures under changing data traffic rates, as previously mentioned (cf., Section 2.1). Each message is assigned a TTL of 1 day, so that node buffers are freed when messages are delivered to their destinations, dropped (according to the FIFO policy) due to buffer overflow, or as a result of TTL exhaustion. For our proposed GReBurD, we have obtained results with a script we have implemented in ONE simulator. With the script, we are able to obtain the burden on each node at different points in time.

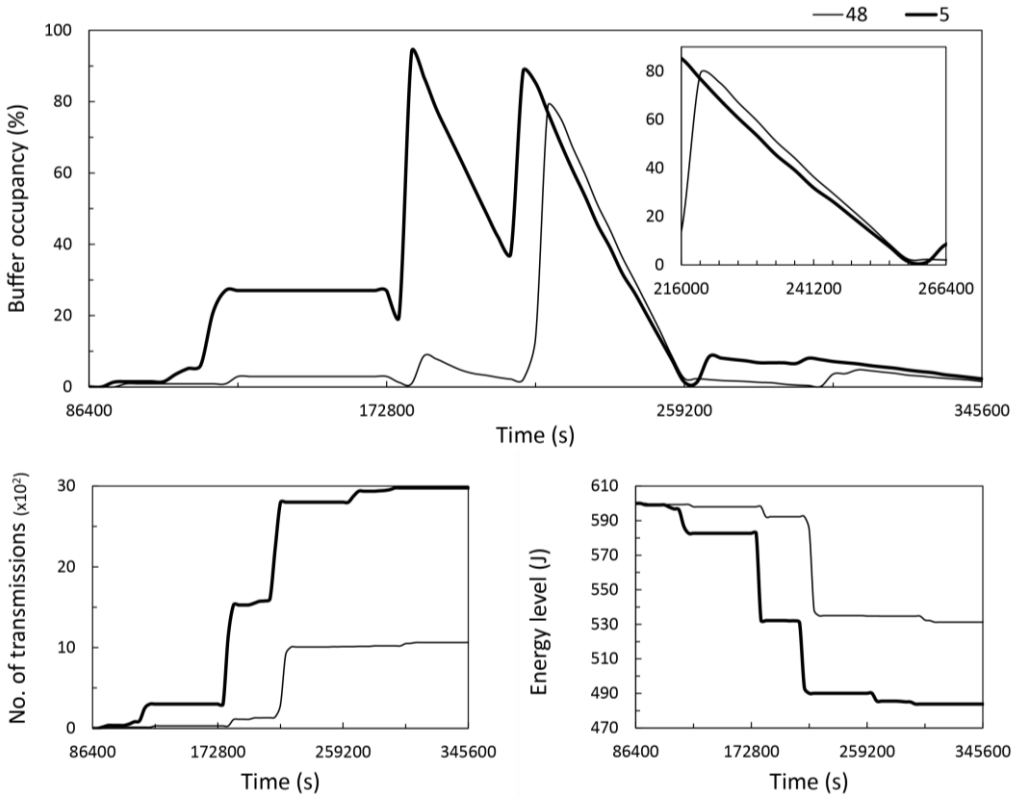
We address the following question: "is there a scenario that could render buffer occupancy inaccurate for determining the relative burden on nodes due to changes in the rate of message generation?" By comparing the results of randomly selected node pairs, some pairs verify that buffer occupancy may not always be an accurate representation of the burden on nodes. In fact, instantaneous buffer occupancy depends on various variables that may not correlate with the relative amount of contribution a node has done in forwarding messages, e.g., TTL, the number of destinations that can be directly encountered, the queuing policy in use, and routing protocol conditions for dropping messages. Figure 1a, 1b and 1c compares three node pairs, Node 39 and 4, Node 41 and 0, and Node 48 and 5, respectively. During the three days of message generation, the second node of each pair (i.e., Node 4, 0 and 5) – which we term as the popular node – transmits more messages (i.e., the sum of sent and received messages), and this is reflected in their energy level. Hence, the less popular node, carrying more messages in its buffer at some point in time does not make its burden exceed that on the popular node, unless this continues for a period of time that is able to compensate for the burden that the popular node has been subjected to. In addition, higher buffer occupancy on the less popular node would mean higher burden only when it has experienced more transmissions – note that at this point in time, it is possible for the node with lower buffer occupancy to be delivering more messages to destinations, hence, freeing its buffer at a faster rate. The figures show that the buffer occupancy on the less popular node exceeds that on the popular node for considerable hours (11, 2 and 12 hours for Figure 1a, 1b and 1c, respectively), during which false positives – in terms of burden – could arise if routing decisions are made solely based on buffer occupancy.



(a)



(b)



(c)

Figure 1. Buffer occupancy, number of transmission and energy level on: (a) Node 39 and 4 (b) Node 41 and 0 (c) Node 48 and 5.

The results in Figure 1 have shown that the instantaneous buffer occupancy may not be able to reflect the amount of transmissions a node has experienced, or the energy it has expended in doing so. By comparing buffer occupancies in the highlighted sections of Figure 1, messages would be directed away from the less popular node and towards the popular node at some point in time. Such approaches try to balance the buffer occupancy of nodes, hence, achieve fairness in terms of buffer occupancy. However, they fail to achieve fairness in terms of the energy consumed in routing, since instantaneous buffer occupancy does not always indicate the actual burden on a node. By observing Figure 1, the current state of the buffer and snapshots of previous states could be used to compute the instantaneous burden. However, there is no guarantee that the snapshot of a previous state would be captured at point in time when the buffer would be able to reflect the actual burden on the node.

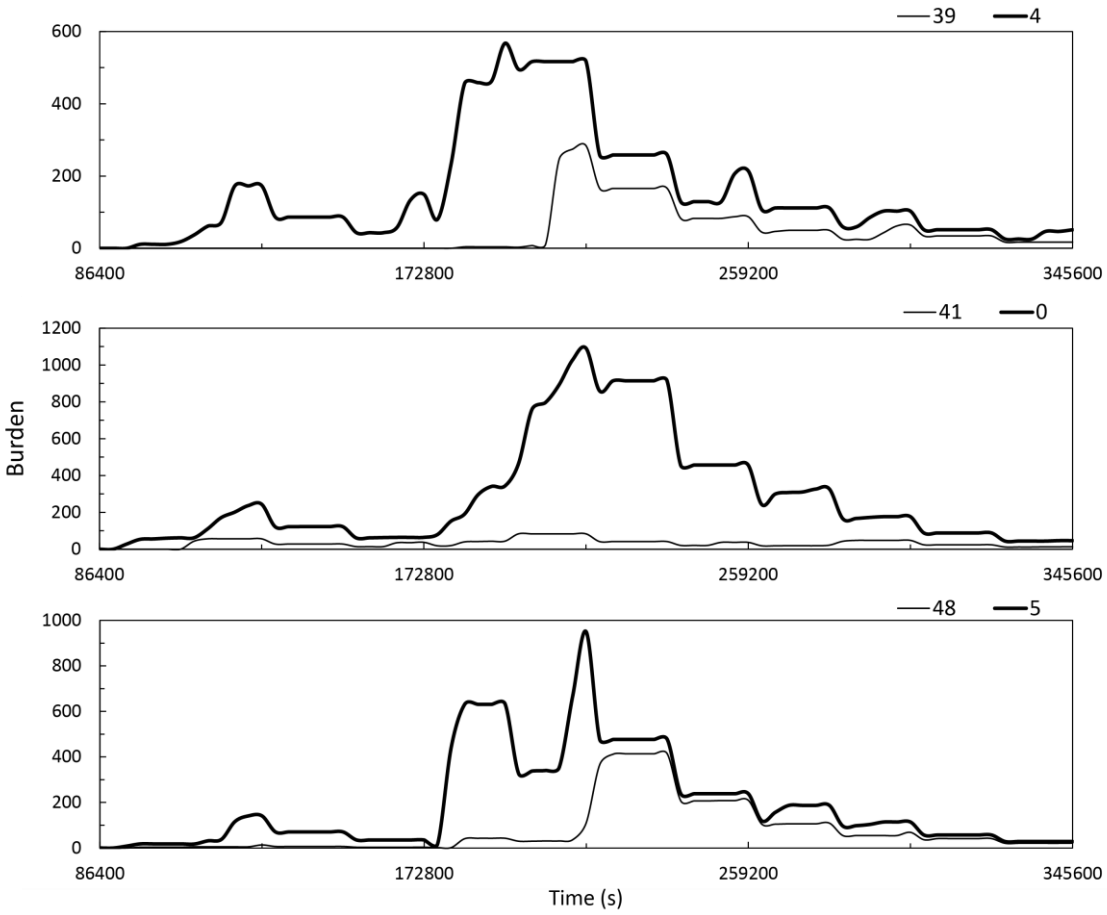


Figure 2. The burden on the nodes according to GReBurD.

The results from this experiment makes it easier to imagine how inferring node burden from buffer occupancy would be a problem in a scenario where every user's device is a potential destination. Popular nodes would then be able to encounter more destinations directly and free up their buffers – lecturers and their students, the bus driver's device and encounters with passenger devices, sales people who serve a lot of customers, etc. Without a steady flow of incoming messages, the buffer occupancy on such nodes may drop at a relatively higher rate, thereby rendering buffer occupancy momentarily inaccurate for determining the relative burden on nodes. To address this issue, our approach considers the number of transmissions within a time frame, and tries

to account for any periods of inactivity between. In other words, our proposed burden measure decreases with reducing number of transmissions and vice versa. Therefore, the burden on a less popular node in Figure 1 will surpass that on its popular counterpart only when the former has done more transmissions and expended more energy. As opposed to inferring the burden from the number of transmissions alone, the reduction in burden when there is less number of transmissions is able to account for periods of inactivity, during which device resources could be replenished (e.g., battery recharge). We believe that our proposed GReBurD is also more suitable in real-life scenarios because it can cope with the dynamicity of node behaviour. Suppose Node A just joined the network. By inferring burden from the number of transmissions alone, it may take a very long time for the burden on Node A to reach that on the other nodes, even if it is a popular node. With our proposed burden measure on the other hand, Node A would be able to catch up in the next counter reset interval. As shown in Figure 1, the burden on the less popular nodes never surpasses that on their popular counterpart, even when the buffer occupancy of the former exceeds that of the latter. This burden corresponds to the number of transmissions and energy expended on both nodes as shown in the figure.

3. Measuring Routing Fairness in PSNs

In this section, we analyse the metrics for fairness, point out shortcomings of using these metrics to evaluate PSNs, and propose a metric specifically for measuring routing fairness in PSNs.

3.1. Existing metrics for routing fairness

In PSNs, the fairness metric measures "evenness" of burden distribution on nodes, and if not even, indicates how far the distribution is from evenness. According to Jain *et al.* (1984), the index should have the following properties in order to give an intuitive understanding of fairness: (i) independent of population size, applicable to any number of nodes; (ii) independent of scale and metric, should give the same results across different units of measurement; (iii) bounded between 0 and 1, a totally fair and a totally unfair distribution should have an index of 1 and 0, respectively; and (iv) continuous, ability to reflect any slight change in distribution.

Various measures are used to determine the level of fairness among nodes in homogeneous networks. Pujol *et al.* (2009) measure fairness by the fraction of nodes that carry out 50% of the total number of forwards in the network. Thus, a larger fraction implies more fairness and vice versa. The metric used by Soelistijanto and Howarth (2012) is the ratio of maximum to mean data traffic seen by nodes in the network, with a lower value implying a more even distribution of traffic among the nodes. In Mashhadi *et al.*'s (2012) evaluation, fairness is given by the coefficient of variation of the total load forwarded by nodes. The index proposed by Mtibaa and Harras (2013) is the difference between the message load distribution given by the forwarding process and the uniform distribution among nodes.

However, none of these measures possesses all the aforementioned desired properties of a suitable fairness index. For instance: continuity, the drawback of the measures used by Pujol *et al.* (2009), and Soelistijanto and Howarth (2012) is similar to that of the max-min ratio (Marson and Gerla, 1982), as they do not reflect slight changes in distribution; and boundedness, the index proposed by Mtibaa and Harras (2013) ranges from positive to negative values and the metric used by Mashhadi *et al.*

(2012) drifts towards $-\infty$ as fairness decreases. As a result, it is not easy to interpret the level of fairness implied by the existing measures. In order to meet account for these criteria, previous work (e.g., Akestoridis *et al.*, 2014; Fan *et al.*, 2014) use Jain *et al.*'s (1984) index (cf., Equation 2) to measure fairness in homogeneous networks.

$$f_A(x) = \frac{[\sum_{i=1}^{n} x_i]^2}{n \sum_{i=1}^{n} {x_i}^2}, \qquad x_i \geq 0 \qquad Equation\ 2$$

In Equation 2, $f_A(x)$ is the fairness index of a protocol A, that distributes x amount of burden to n nodes, such that the i^{th} node receives a burden x_i.

Table 2. Fairness of distributing a total burden of 10 to two nodes in a homogeneous network according to Jain *et al.* (1984).

	Algorithm and burden distribution					
	A	**B**	**C**	**D**	**E**	**F**
Node 1	10	9	8	7	6	5
Node 2	0	1	2	3	4	5
Fairness (%)	50	60.98	73.53	86.21	96.15	100

From Table 2, algorithm A is totally unfair since node 1 bears the entire burden in the network and F is totally fair since the total burden is equally distributed between both nodes. Thus, $f_A(x)$ is expected to be 0 and $f_F(x)$ is expected to be 1, which corresponds to a fairness of 0% and 100%, respectively. However, Jain *et al.*'s (1984) index assigns a fairness of 50% to algorithm A – instead of 0% – which is not a suitable interpretation of fairness in PSNs. According to Jain *et al*. (1984), "a distribution algorithm with a fairness of 0.1 means that it is unfair to 90% of the users". This implies that algorithm A is unfair to only 50% of the nodes, hence, results in a 50% fairness. In terms of evaluating fairness in a PSN, however, this scenario would be better interpreted as total unfairness, which means a 0% fairness. To account for this, we propose a new metric for measuring PSN routing fairness in Section 3.2.

3.2. Proposed metric for routing fairness

To illustrate our proposed fairness metric, consider a scenario of six fictitious forwarding algorithms, A to F, with 4 nodes each, n1 to n4, that are burdened at different extents for each forwarding algorithm (cf., Table 3). The standard deviation from the mean burden for a forwarding algorithm, a, is given by Equation 3.

$$\beta_a = \sqrt{\sum_{i=1}^{n} \left(b_{a,i} - B_a\right)^2 / n} \qquad Equation\ 3$$

Where n is the total number of nodes, $b_{a,i}$ is the burden experienced by node i running on algorithm a, and B_a is the mean burden experienced by a node under algorithm a. Then $\beta_{a,max}$, the maximum possible standard deviation for algorithm a (i.e., the standard deviation if only one node bears all the burden under algorithm a, which is also the least fair scenario, such as algorithm F in Table 3), is given by Equation 4.

$$\beta_{a,max} = \sqrt{\left[\left(b_{a,total} - B_a\right)^2 + {B_a}^2(n-1)\right]/n} \qquad Equation\ 4$$

Where $b_{a,total}$ is the total burden, i.e., sum of the burdens experienced by every node in the network. As given by Equation 5, the ratio between β_a and $\beta_{a,max}$ indicates the unfairness of algorithm a.

$$U_a = \frac{\beta_a}{\beta_{a,max}} \times 100 \qquad Equation\ 5$$

$$F_a = 100 - U_a \qquad Equation\ 6$$

If every node experiences equal burden for algorithm a, F_a in Equation 6 will become 100% (e.g., A in Table 3). Likewise, if only one node bears all the burden, F_a will become 0% (e.g., F in Table 3). F_a also decreases accordingly, the less evenly the total burden is distributed among the nodes (e.g., B to E in Table 3). Therefore, F_a can be used to measure the fairness of a forwarding algorithm, a, in terms of the burden on nodes.

Table 3. Six fictitious forwarding algorithms for describing the working principle of the proposed fairness metric and how it interprets fairness.

Forwarding algorithm (a)	Nodes and burden				β_a	Fairness; F_a (%)
	n1	n2	n3	n4		
A	12	12	12	12	0	100
B	13	12	12	11	0.7	96.6
C	24	8	8	8	6.9	66.7
D	32	8	4	4	11.7	43.9
E	47	1	0	0	20.2	2.8
F	48	0	0	0	20.8	0

3.3. Evaluation of proposed metric for routing fairness

Our proposed fairness metric is applicable to any number of nodes, gives the same results across different units of measurement, results in 0 and 1 for a totally fair and a totally unfair distribution, and has the ability to reflect any slight change in distribution. Here, we compare the performance of our proposed fairness metric with Jain *et al.*'s (1984), which is the only one among the existing metrics that possess all the stated desirable properties of a fairness metric. As shown in Table 4, we allocate 10 items to two nodes according to 6 different allocation schemes. In a PSN scenario, routing fairness should be 100% when the two nodes experience equal burden, and both fairness metrics are able to reflect this for Scheme 6. On the other hand, burdening only one node in a PSN should result to a 0% fairness, which we expect for Scheme 1. With Jain *et al.*'s (1984) metric, Scheme 1, the most unfair allocation scheme, gives a fairness of 50%, while our proposed metric gives a fairness of 0%. With this behaviour, it is easier to give a better interpretation of the level of fairness implied by our proposed fairness metric. As shown in Table 4, our proposed fairness metric is able to reflect a totally unfair and a completely fair allocation of burden in PSNs. The fairness of a forwarding algorithm can be computed from the burden measure we proposed and evaluated in Section 2, using the instantaneous burden on each node.

Table 4. Fairness of different schemes for allocating 10 items to two nodes.

Allocation scheme	Allocated items		Fairness metric (%)	
	Node A	Node B	Proposed	Jain *et al.*'s
1	10	0	0	50.0
2	9	1	20	61.0
3	8	2	40	73.5
4	7	3	60	86.2
5	6	4	80	96.2
6	5	5	100	100.0

The difference between the two metrics is in their interpretation of minimum fairness. First, Jain *et al.*'s (1984) metric considers the entire population. Hence, the extent of fairness interpreted when only one node bears all the burden changes with the total population (cf., Figure 3). Although reasonable, fairness never reaches 0%, and one is forced to bear in mind the total population in order to fully understand the interpretation – since the most unfair scenario could be represented by different values. Second, the condition for Jain *et al.*'s (1984) metric to give 0% fairness is when the numerator of Equation 2 equals 0. This only occurs when negative burden values are considered, i.e., 10 and -10 gives a 0% fairness in this case. However, since burden values are usually non-negative, Jain *et al.*'s (1984) metric may require them to undergo specific normalization processes in order to be bounded between 0 and 1 (e.g., representing 10 and 0 by 5 and -5, respectively).

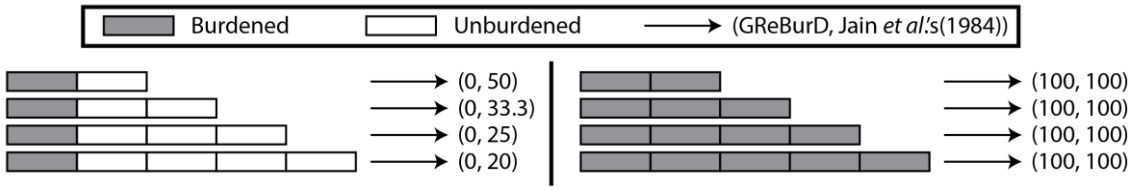


Figure 3. Fairness according to our proposed metric and Jain *et al.*'s (1984).

Considering that instantaneous burden on nodes changes over time, burden at different points in time may have to be averaged for each node in order to obtain the actual fairness of a routing algorithm. Using this method and a granularity of 3600 seconds, we are able to compute the fairness of the scenario in Section 2.3 with our proposed metric and Jain *et al.*'s (1984) metric as 95.44% and 44.54%, respectively.

4. Conclusions and Future Work

In a Pocket Switched Network (PSN), portable handheld devices, such as smartphones and tablets, store messages, carry them through user movement, and forward them to other devices when a communication opportunity arises. The network thus, depends on the willingness of users to participate and share their devices as routers. Due to the versatility of portable handheld devices, the availability of required resources is limited. Furthermore, users may not be willing to shed all their resources on behalf of the network. Therefore, it is important that PSN protocols be as fair as possible by sparingly utilizing the resources available on each node instead of subjecting most of the routing burden to only a few set of popular nodes. Overlooking fairness results in drastic resource consumption on popular nodes, and may eventually lead to user dissatisfaction, withdrawal, and performance degradation of the network.

In order to ensure fairness in PSN routing, network designers need to be able to estimate the burden that has been impacted on nodes, utilize this knowledge to provide an acceptably fair utilization of node resources, and evaluate the level of fairness achieved. Existing approaches for ensuring routing fairness rely on buffer occupancy to indicate the level of burden on nodes. Unfortunately, as experiments in Section 2 have revealed, buffer occupancy may not always be able to reflect the amount of burden that a node has been subjected to. In this regard, we have proposed GReBurD, a mechanism that estimates the burden on PSN nodes, and evaluated its performance in ONE simulator. GReBurD is simple yet effective, as it is non-scenario specific and better infers the actual burden on a node as compared with existing approaches.

We also analysed existing metrics for measuring fairness in the distribution of burden among nodes. We identified that most of the metrics fail to satisfy the desirable properties of a fairness metric, while the most accepted metric leaves room for improvement in terms of interpreting the level of fairness implied. In this regard, we have proposed a new metric with which network designers can evaluate the fairness of PSN forwarding algorithms. The proposed metric possesses all the desirable properties of a fairness metric, and also gives a better interpretation of the level of fairness implied in PSNs.

Future work includes a distributed version of GReBurD that can locally estimate the relative burden on a node, without requiring synchronization between devices. Future work also includes investigating the impact of different reset intervals on the performance of GReBurD. For the future version, we also consider a means of resetting the transmission counter at optimum periods detected from device activity history, instead of selecting fixed intervals. We also have planned to evaluate the GReBurD in a real-life scenario.

Acknowledgment

Thanks are due to Fundação de Amparo à Pesquisa do Estado de Goiás (FAPEG) and PNPD/CAPES.

References

Akestoridis, D., Papanikos, N., and Papapetrou, E. (2014). "Exploiting social preferences for congestion control in opportunistic networks," 2014 IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Larnaca, pp. 413-418. DOI: 10.1109/WiMOB.2014.6962204.

Amah, T., Kamat, M., Moreira, W., Abu Bakar, K., Mandala, S., and Batista, M. (2016). Towards next-generation routing protocols for pocket switched networks, Journal of Network and Computer Applications, Volume 70, pp. 51–88. DOI: 10.1016/j.jnca.2016.05.011.

Ekman, F., Keränen, A., Karvo, J., and Ott, J. (2008). Working Day Movement Model. MobilityModels'08, Hong Kong SAR, China.

Fan, X., Li, V., and Xu, K. (2014). "Fairness Analysis of Routing in Opportunistic Mobile Networks," In IEEE Transactions on Vehicular Technology, vol. 63, no. 3, pp. 1282-1295. DOI: 10.1109/TVT.2013.2282341.

Grasic, S., Davies, E., Lindgren, A., and Doria, A. (2011). The evolution of a DTN routing protocol—prophetv2. In: Proceedings of the 6th ACM workshop on challenged networks, CHANTS '11. ACM, New York, pp 27–30. DOI:10.1145/2030652.2030661.

Grundy, A., and Radenkovic, M. (2010). "Promoting congestion control in opportunistic networks," 2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications, Niagara Falls, ON, pp. 324 330. DOI: 10.1109/WIMOB.2010.5645048.

Jain, R., Chiu, D., and Hawe, W. (1984). "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," Digital Equipment Corporation, Tech. Rep. DEC-TR-301.

Keränen, A., Ott, J., and Kärkkäinen, T. (2009). The ONE simulator for DTN protocol evaluation. Simutools '09, Proceedings of the 2nd International Conference on Simulation Tools and Techniques. Rome, Italy. No. 55. DOI: 10.4108/ICST.SIMUTOOLS2009.5674.

Marson, M., and Gerla, M. (1982). "Fairness in Local Computing Networks," In Proceeding of IEEE ICC.

Mashhadi, A., Mokhtar, S., and Capra, L. (2012). Fair content dissemination in participatory DTNs, Ad Hoc Networks (Special Issue on Social-Based Routing in Mobile and Delay-Tolerant Networks), Volume 10, Issue 8, November 2012, Pages 1633–1645. DOI: 10.1016/j.adhoc.2011.05.010.

Mtibaa, A., and Harras, K. (2013). Fairness-related challenges in mobile opportunistic networking, Computer Networks, Volume 57, Issue 1, pp. 228-242. DOI: 10.1016/j.comnet.2012.08.019.

Muchnik, L., Pei, S., Parra, L., Reis, S., Andrade, J., Havlin, S., and Makse, H. (2013). Origins of power-law degree distribution in the heterogeneity of human activity in social networks. Scientific Reports 3: 1783. DOI: 10.1038/srep01783.

Orlinski, M., and Filer, N. (2012). Quality distributed community formation for data delivery in pocket switched networks. In Proceedings of the Fourth Annual Workshop on Simplifying Complex Networks for Practitioners (SIMPLEX '12). ACM, New York, NY, USA, pp. 31-36. DOI: 10.1145/2184356.2184365.

Pujol, J., Toledo, A., and Rodriguez, P. (2009). "Fair Routing in Delay Tolerant Networks," INFOCOM 2009, IEEE, Rio de Janeiro, pp. 837-845. DOI: 10.1109/INFCOM.2009.5061993.

Soelistijanto, B., and Howarth, M. (2012). "Traffic distribution and network capacity analysis in social opportunistic networks," 2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Barcelona, pp. 823-830. DOI: 10.1109/WiMOB.2012.6379171.

Tsvetovat, M., and Kouznetsov, A. (2011). Centrality, Power, and Bottlenecks. Social Network Analysis for Startups. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Remote Routing Approach to Restricted Devices in MANETs

Rodrigo Melo[1], Rafael R. Aschoff[2], Djamel Sadok[1], Eduardo Feitosa[3]

[1]Center for Informatics – Federal University of Pernambuco (UFPE)
Pernambuco – Brazil

[2]Pernambuco Federal Institute of Education, Science, and Technology (IFPE)
Palmares, Brazil.

[3]Federal University of Amazonas
Manaus – Brazil

{rodrigodma,jamel}@gprt.ufpe.br, rafael.roque@palmares.ifpe.edu.br,
∈feitosa@icomp.ufam.edu.br

Abstract. *Emergency rescue communication systems are designed to provide ubiquitous collaboration among mobile devices without the need for a fixed infrastructure by using mobile ad hoc networks (MANETs). In emergency and rescue operations, MANETs are naturally formed by devices with different processing and communication capabilities. In this scenario, low power devices may become overwhelmed with the control overhead and resulting additional processing required to provide reliable communications among these heterogeneous parties. In this paper we propose a strategy where resource constrained devices can offload part of the routing process to more capable devices participating in the network. Our proposal have been tested and validated in different scenarios. The results show how the proposed approach can successfully reduce the network overhead without significantly reducing the routing process efficiency.*

1. Introduction

The increasing interest in using mobile devices to establish ad-hoc communication systems (MANET) has enabled the development of a broad range of applications, particularly with the advent of new wireless access technologies for connectivity, such as 3/4G, LTE, and WiMax. However, these new applications typically assume that MANETs are a set of homogeneous devices, where each node has the same capabilities, which is usually unrealistic.

MANETs support several types of mobile devices and this is one of its success factors. This heterogeneity allows great flexibility, however, when devices with limitation in battery or processing power join the network new problems arise. In this paper, we concentrate on the routing process issues for restricted devices. For such devices, participating in large networks may not be possible due to the routing process that can be very resource consuming. For example, during the routing process, topology information must reach the whole network, which generates a great amount of overhead that can compromise the performance of these special devices.

The most popular strategy to address this problem is to reduce the routing protocol overhead, particularly, control packets that consume network bandwidth and other

resources. Works like [Pei et al. 2000] and [Younis et al. 2002] have been proposed with solutions focusing in performance improvements. However, these solutions aim at the overall network performance, while our concern is on allowing restricted devices in the networks. We focus on techniques to save the resources of restricted devices during the routing decision process so they can participate in the network.

This paper proposes the Distributed Remote Routing (DRR), a strategy to offload part of the routing decision process of restricted devices to more capable devices. Developed to work on networks running the HTR protocol [Souto et al. 2012], we introduce a new categorization of HTR nodes: the HTR-Lite (HTR-L) nodes, which indicates the restricted devices and the HTR-Outsourcing Router (HTR-OR), the more capable devices that will be responsible for helping the HTR-L nodes in the routing process.

2. HTR Overview

HTR is a routing protocol for MANETs that utilizes a 2.5 cross layer scheme [Souto et al. 2012]. It abstracts multiple and heterogeneous interfaces and constructs a self-organized heterogeneous communicating ad hoc network. An HTR node may have many interfaces, with similar or different technologies such as Wi-Fi, Bluetooth, WiMAX and LTE, but it has only one IP address.

HTR is based on the OLSR protocol [Clausen and Jacquet 2003] and similarly includes HELLO and Topology Control (TC) as control messages and defines a special node, the multi-point relay or MPR, for the control of traffic flooding. From a HELLO message, a mobile node receives information about its immediate, 2-hop neighbors and selects MPRs accordingly. A TC message originates at an MPR node announcing who has selected it as an MPR. In contrast to OLSR, HTR uses additional metric based on link quality information and node device capabilities to choose MPR nodes. Called HTRScore, the HTR cost metric is defined considering factors such as the awareness of link conditions and power efficiency in order to perform path computation.

The HTRScore formula can be seen at (1).

$$HTRScore(i,j) = \frac{e_{i,j}^{\alpha}}{(1-\rho_{i,j})^{\beta}} * \frac{E_{\gamma}^{i}}{R_i^{\Theta}} \tag{1}$$

Where i is the source node; j is the destination neighbor; $e_{i,j}$ is the transmission energy required for node i to transmit an information unit to its neighbor j; $\rho_{i,j}$ is the probability to lose a packet sent from i to j; R_i is the residual energy of node i; and E_i is the initial battery energy of node i.

The symbols α, β, γ and θ represent non-negative weighting factors for each described parameter. Note that if all weights are equal to zero, then the lowest-cost path is the shortest path, and if only γ, and θ are equal to zero, then the lowest cost path is the one that will require the least energy consumption, considering retransmission or not, regarding the value of β. If γ is equal to θ then normalized residual energy is used, while if only θ is equal to zero then the absolute residual energy is used. In case all three parameters α, γ and θ are equal to zero, then only the paths with best link stability are emphasized.

Two main modules compose the HTR framework: the bootstrap module and the routing module. The bootstrap module is responsible for the start-up and configuration of

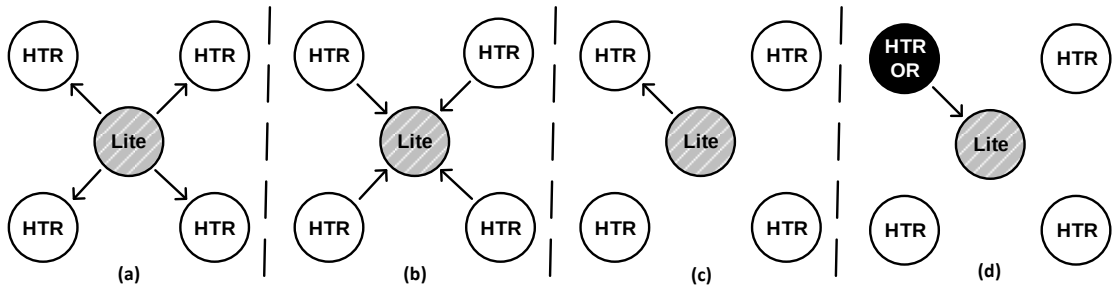


Figure 1. HTR Lite steps of the remote routing solution

a node (i.e. assignment of IP address and link layer adaptive configuration). The routing module manages the routing table and packet forwarding. It uses the Dijkstra Algorithm to perform path computation, however, the edges of the network graph have a weight equals to the HTRScore described above. Specific details regarding these modules can be found in [Souto et al. 2012].

3. Distributed Remote Routing

On a HTR-running network, each node is responsible for building its own routing table based on information received via the control messages. Each node knows its neighbors and is capable of deciding the best path to reach a destination. As introduced above, this may constitute a restricting factor for some devices.

The Distributed Remote Routing (DRR) is our approach to relax this restriction. It was developed to save resource from these nodes by offloading routing decisions (routing table calculation) to other devices within the network.

There are three fundamental roles in the DRR process as described below.

- **HTR nodes:** nodes that run native HTR protocol. They send and receive HTR control messages (HELLO and TC) and build their own routing table.
- **HTR-Lite nodes:** restricted devices. HTR-lite nodes do not send HELLO nor TC messages and do not act as routers. In other words, HTR-Lite nodes will not appear as a valid entry on the forwarding table of the network nodes. HTR-lite nodes mount their routing table by sending a table request to HTR-OR nodes.
- **HTR-OR nodes:** HTR-Outsourcing Router nodes are responsible to provide the routing table information requested by HTR-Lite nodes. There is no limit regarding the number of HTR-OR nodes participating in the network.

We embedded the DDR in the HTR framework but, as expected, DDR has its own set of messages and control states. Figure 1 illustrates the protocol operation.

As shown in the figure, the HTR-Lite initiates the operation as soon as it joins the network (managed by the bootstrap module). The node sends a broadcast request (discovery message) in search for a HTR-OR (a). When a HTR-OR node receives a discovery message (b), it replies with its HTRScore directly to the requesting node (HTR-Lite).

After having received replies from one or more HTR-OR nodes, the HTR-lite node uses the informed HTRScores as criteria to choose the node that will act as its HTR-OR. Having selected the HTR-OR the HTR-Lite node sends a Route Request message

to it (c). Finally, upon receiving a route request message, the HTR-OR node sends the Routing Table Reply message to the requesting node (d).

It is important to emphasize that there is no significant increase of resource consumption by the HTR node that becomes HTR-OR. This is because it will only send its already calculated routing table to the requesting HTR-lite node.

Since the HTR-OR only sends its own routing table, the HTR-Lite needs to perform minor adjustments to such table to adapt it to the point-of-view of the HTR-Lite itself. The algorithm to make this adjustments was designed to consider the cost of manipulating the route data without incurring in too much resource consumption. Firstly, during the second step of the remote routing process (see Figure 1), not just the elected HTR-OR but all nodes that replied to the Discovery Message are added to the neighbor table. Then, after receiving the routing table from a HTR-OR, the insert route table algorithm is started. The procedure is described in Algorithm 1. Simply put, the algorithm verifies the received routing table (line 4) and for every advertised entry (line 6) that has not already been added (line 7), it changes the next-hop field to the selected HTR-OR (line 8).

Algorithm 1 Insert Route Table Algorithm

```
1:  procedure INSERTTABLE(routeOR, addrOR)
2:      size ← routeOR.size()
3:      index := 0
4:      table ← getRouteTable()
5:      while index < size do
6:          route ← routeOR.get(index)
7:          if table.hasNoEntry(route) then
8:              route.updateNextHop(addrOR)
9:              table.insert(route)
10:         end if
11:         index := index + 1
12:     end while
13: end procedure
```

In order to comply with the unpredictable nature of the mobile environment, the received routing table is set to expire after a period and the whole process starts again. Figure 2 shows the state machine of the lite node in this proposed DRR protocol.

As shown in the figure, there are four states in the lite node: Start, Wait Reply, Wait Route Table, and Route Table Received. The Start state represents the initial phase, where the lite node wishes to initiate the remote routing process. After sending a Request Message, the lite node goes into the Wait Reply state. If the lite node receives no reply, it will constantly attempt to send another request after a timeout interval. If a reply is received, the node sends a Select Route message and goes into the Wait Route Table state. Similarly, at this state the lite node will attempt to retry to send the Select Route message after a timeout occurs, but only for a limited amount of time. If the route table is received, the node ends its configurations process; otherwise, it goes into the Wait Reply state to start the process again. Important to say that this state machine is always running when entry in network or when the table received expires.

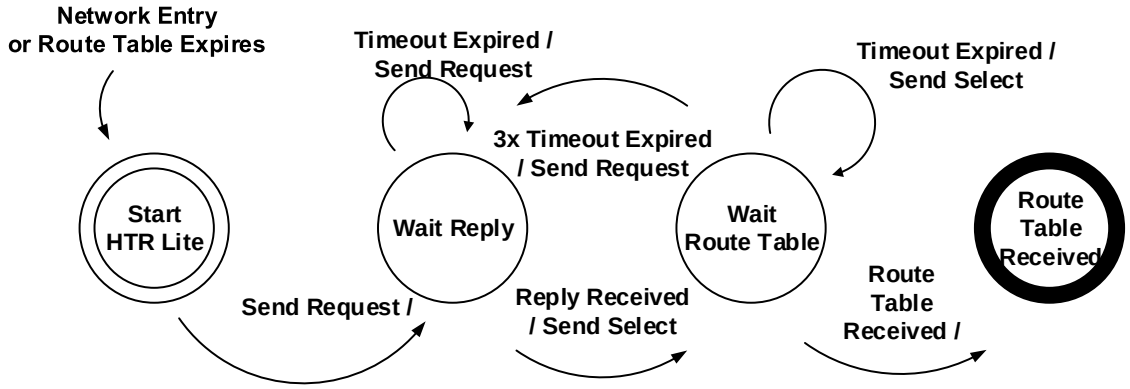


Figure 2. State Machine of the remote routing protocol for the Lite node

The server side works as a stateless process, as describe in Algorithm 2. The server continually listen to received messages (lines 2-3). When the server receives a Discovery Message (line 5), it sends back a Reply Message (line 6). If the server receives a Request Message, it sends a Route Reply Message.

Algorithm 2 Server side remote routing process routine

```
1:  procedure RECEIVEMESSAGE
2:      while true do
3:          message, srcAddress ← receiveMessage()
4:          type ← message.getType()
5:          if type = DISCOVERY then
6:              sendScore(srcAddress)
7:          else if type = REQUEST then
8:              sendReply(srcAddress)
9:          end if
10:     end while
11: end procedure
```

4. Evaluation Methodology

In order to evaluate the performance of the proposed protocol, we have used different scenarios and metrics. This section describes the methodology we used to perform the evaluation the proposed protocol.

4.1. Metrics

Given that the proposed routing protocol does not change the network behavior, apart from the process to obtain the routing table, only a few network metrics would have a possible change on their measurable values comparing to the standard approach. We chose to evaluate the routing delay, and message overhead, which are better described below.

Routing Delay. The amount of time a HTR-lite node takes to obtain the routing table shared by one HTR-OR, after the complete remote routing process was performed. More specifically, it constitutes the time between the discovery message and the routing

table reply message. As illustrated by Figure2, the process includes intermediate steps, such as the select and reply messages as well as possible timeouts and reattempts.

Message Overhead. The message overhead is a metric to measure the amount of network bandwidth save while using the proposed remote routing protocol. Nodes running the remote routing do not participate in the forwarding of messages nor send HELLO or TC messages, which saves resources. This metric compares the amount of traffic generated by networks with and without the remote routing protocol.

These metrics were evaluated using different scenarios, as described in the next subsection.

4.2. Scenarios

We decided to use two circular topology scenario to evaluate our work. The choice of such a simple scenario is motivated by the fact that our focus is on the performance of the HTR-Lite itself, since the approach has a negligible impact on the other components of the network.

For the first scenario, we configured one HTR-lite surrounded by a group of HTR-OR. We then vary the number of HTR-OR surrounding the HTR-lite (Figure 3). The objective of this scenario is to illustrate the viability of using the DRR protocol regarding the number of nodes available to share routing tables.

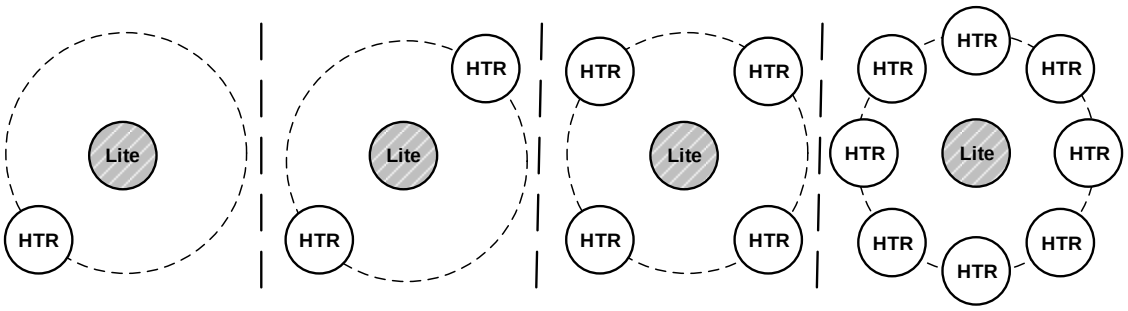


Figure 3. Topology of the first scenario

The second scenario is the inverse of the first one. In other words, the second scenario is configured with a HTR-OR surrounded by a group of HTR-Lite (Scenario 2). This scenario is important to detect the impact and scalability of the solution by detecting the number of HTR-lite nodes that overloads or decrease significantly the efficiency of the HTR-OR node. This scenario is illustrated in the Figure 4.

For both scenarios, in addition to varying the number of surrounding nodes, the mobility mode of the nodes (static or mobile) of the edge can be configured. These scenarios and metrics were implemented in NS-3 simulator which made it possible to arrange a circular topology to run the experiments a hundred times, for each scenarios. These results of the experiments executed with the presented scenarios are analyzed and discussed in the next section.

5. Results and Discussion

As described in Section 4, we decided to evaluate our proposal against different network topologies and performance metrics. This section presents our finds and discusses the

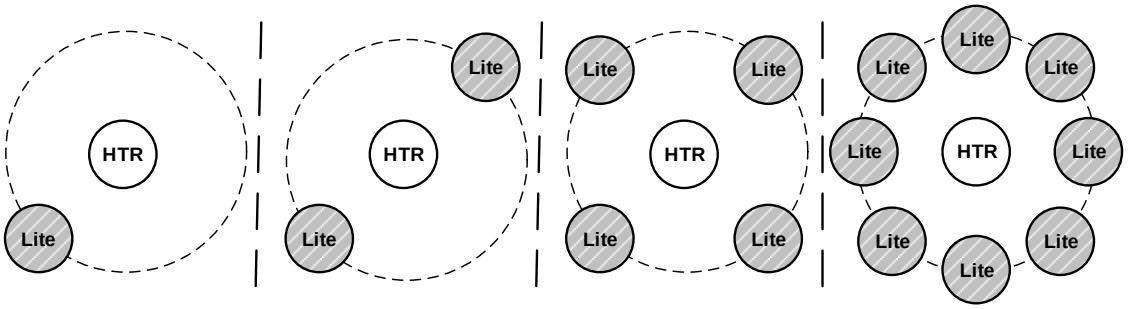


Figure 4. Topology of the second scenario

applicability of the approach giving the results.

We first present the results of the two metrics (Routing Delay and Message Overhead) for the scenario with one HTR-lite surrounded by a group of HTR-OR (Scenario 1). Next, and similarly, we present the results in the case where a HTR-OR is surrounded by a group of HTR-Lite (Scenario 2).

In order to present the results of the Message Overhead metric, we used a marked line chart with two data series. The data series represents the static and dynamic topologies. The x-axis presents the number of surrounding nodes (HTR-OR in case of Scenario 1 and HTR-Lite in case of Scenario 2). The y-axis shows the Message Overhead Reduction (MOR), which represents the percentage of reduction in the overhead when using the DRR protocol. More specifically, we have $MOR = 1 - \frac{O_{ddr}}{O_{htr}}$, where O_{ddr} is the global routing control messages overhead when using the DDR and O_{htr} is the global routing control messages overhead when using only the HTR.

On the other hand, to show the results of Routing Delay metric it was necessary to use two kind of graphs. The first one is a dispersion chart, which represents a grouping routing delay values of each scenario during one simulation timeline. This chart helps to visualize the behavior of the metric during the lifetime of a Lite node in the network. The second chart used to illustrate the routing delay results was a boxplot chart, which shows the minimum, maximum and average value of routing delay metric in each scenario. This representation goes to show the most limiting values achieved by this metric in each simulated scenario.

5.1. Scenario 1

Figure 5 illustrate the results for the Message Overhead metric of Scenario 1 when varying the number of surrounding nodes (HOT-OR) from one to 64.

As shown in Figure 5, the scenario with just two nodes in the network (one HTR-Lite and one HTR-OR) the Message Overhead Reduction was close to 30% for the static mobility model and slightly above that when the HTR-OR can freely move. The bandwidth save or reduction in the number of control messages can be observed with up to four or 16 HTR-OR nodes for the static and mobile configuration respectively. As previously explained, the DDR does not send the Hello or TC messages, which explains the overall reduction of the control sent in the network up to the above commented points. Since the wireless medium is shared amongst the participant nodes, the reduced number of messages sent over the network means a potentially better overall network performance.

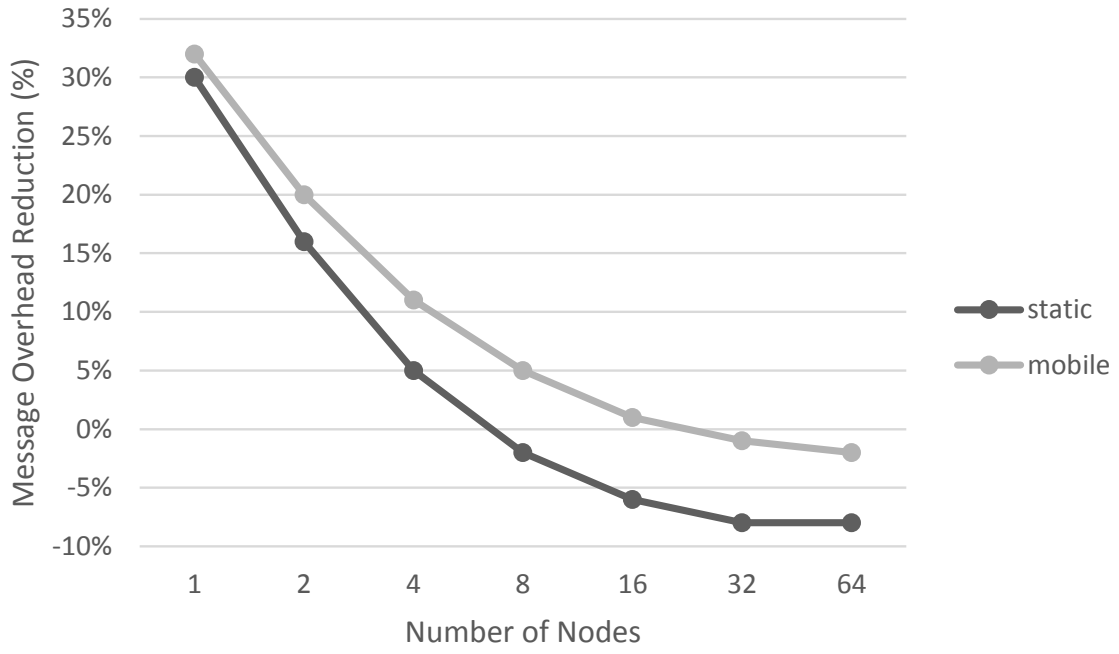


Figure 5. Overhead of messages of scenario 1

As the number of HTR-OR grows larger though, the DDR actually increases the total number of control messages generated in the network. We have to keep in mind, however, that this is observed only because we are increasing the number of HTR-OR while maintaining a single HTR-Lite. In other words, while every new additional HTR-OR have to deal with some additional control messages required by the DDR, there is only one node saving resources. Moreover, this increase would only be perceived in the neighborhood of the HTR-Lite node. Finally, this additional overhead generated in the neighborhood for larger network stabilizes between 32 and 64 nodes.

The Figure 6 shows the results of the Routing Delay metric for the static configuration of our Scenario 1. More specifically, the figure illustrates the observed pattern for a single node in different network densities where we took the sequential readings of the routing delay for node one in a single simulation and varied the number of HTR-OR nodes.

During our preliminary analysis we found out unexpected moments where the delay were much larger then the usual collected data. We first thought that the mobility of the nodes could be causing this larger delays, but in the scenario with fixed nodes we could observe the same pattern. Next, we thought that some missing packets were causing these high delays. We found out, however, that even when everything went smoothly with our routing protocol, there would be instances of large delays.

As can be observed in the figure, independently of the network size, the time series of the routing delay of the node presented cyclical peaks. It turned out that the lower communication layers were causing this unexpected behavior. More precisely, the translation process between the network layer addresses into link layer addresses performed by the Address Resolution Protocol (ARP) was the culprit. The entries in the ARP cache con-

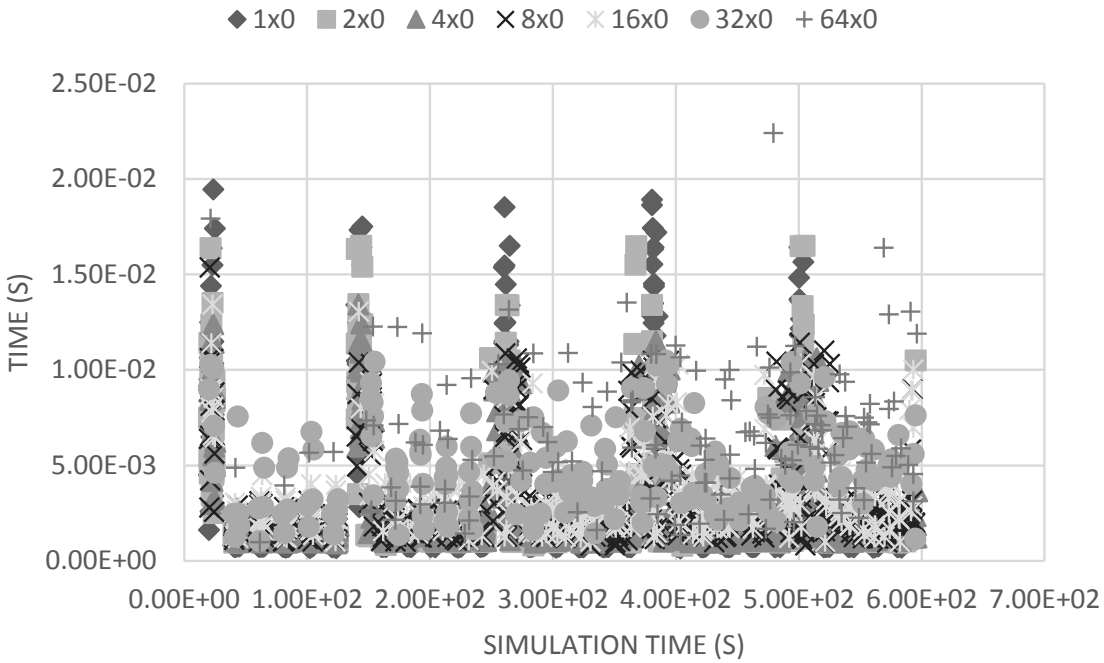


Figure 6. Routing delay of static scenario 1

taining the map between the IPv4 addresses and MAC addresses were expiring or absent, thus requiring an ARP request and reply messages prior to using the IPv4 address.

Giving the specific behavior observed for the static configuration and shown in Figure 6, we decide to verify if the pattern would remain the same when the nodes are configured to freely move. As shown in Figure 7, the same pattern can be observed early on the simulation, but when the nodes are farther apart, the behavior becomes a bit more chaotic. This behavior is to be expected due to the randomness introduced by the mobility of the nodes.

Overall, we can argue that the DDR protocol behaved quite well under the stressed circumstances created for the Scenario 1. It reduced the network overhead in the vicinity for low-density networks, while not greatly increasing the overhead for highly density networks. Moreover, we have to remember that we are potentially including nodes in the network that otherwise would not be able to participate.

5.2. Scenario 2

As previously presented, the Scenario 2 constitutes the inverse topology of the Scenario 1, with a central HTR-OR surrounded by a varying number of HTR-Lite nodes. Figure 8 illustrate the results for the Message Overhead metric of Scenario 2 when varying the number of surrounding nodes (HOT-Lite) from one to 64.

Contrary to what was observed in Figure 5, the Message Overhead Reduction increases as the network topology becomes denser. If in the previous scenario the ratio between HTR-Lite and HTR-OR decreases as the network becomes denser, causing a decrease in the Message Overhead Reduction; the reverse is observed for Scenario 2. In other words, the number of HTR-Lite nodes are increasing relatively to the single HTR-

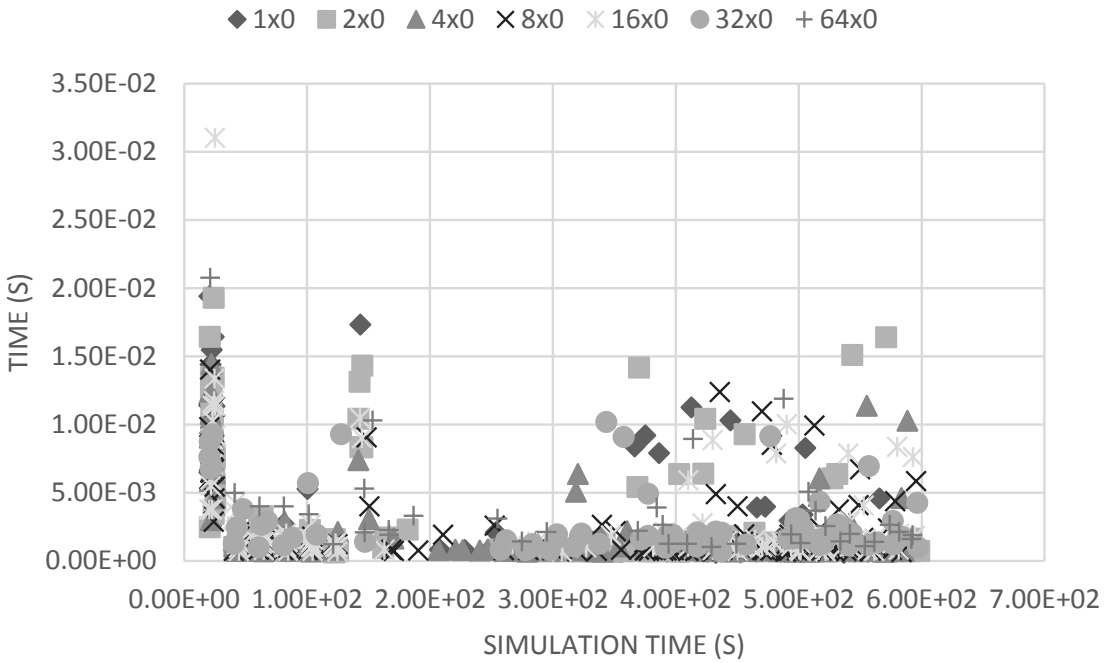


Figure 7. Routing delay of mobile scenario 1

OR and, thus, since the HTR-Lite does not send Hello or TC messages, it was to be expected the reduction of the number of control messages in the overall.

For the second scenario, we were also interested in investigating the unusual pattern (pikes) identified during our experiments. Once more, Figure 9 illustrates a behavior of a single node in different network densities where we took the sequential readings of the routing delay for such node. As shown in the figure, the same pattern can be observed, but it is clearer. The reason for this more deterministic behavior can be explained due to the fact that the HTR-Lite sends messages in a more expected and cyclic behavior.

In the same setup but with mobile nodes (Figure 10) the results are similar to what we found in Scenario 1. Once more, it is to be expected, since the the nodes may become out of reach and latter join again in the same cell, thus presenting a more random behavior.

Our results in both Scenario 1 and Scenario 2 show that our approach does note incur in significant impact on the network. It may be important to note that in all scenarios and configurations, the routing delay did not go above three milliseconds. Such value may not be practical in real environment, giving the additional time required by software and hardware related routines, but comparing with the values collected by the standard approach in the same simulated environment it proves to be satisfactory. Both Routing Delay and Network Overhead are maintained in moderate levels throughout the experiments, while we are able to ensure that a new class of restricted nodes become part of the network.

6. Background

Offloading is the ability to delegate the obligation over some task from one entity to another. It is usually employed to either free one processor that is already fully loaded or to

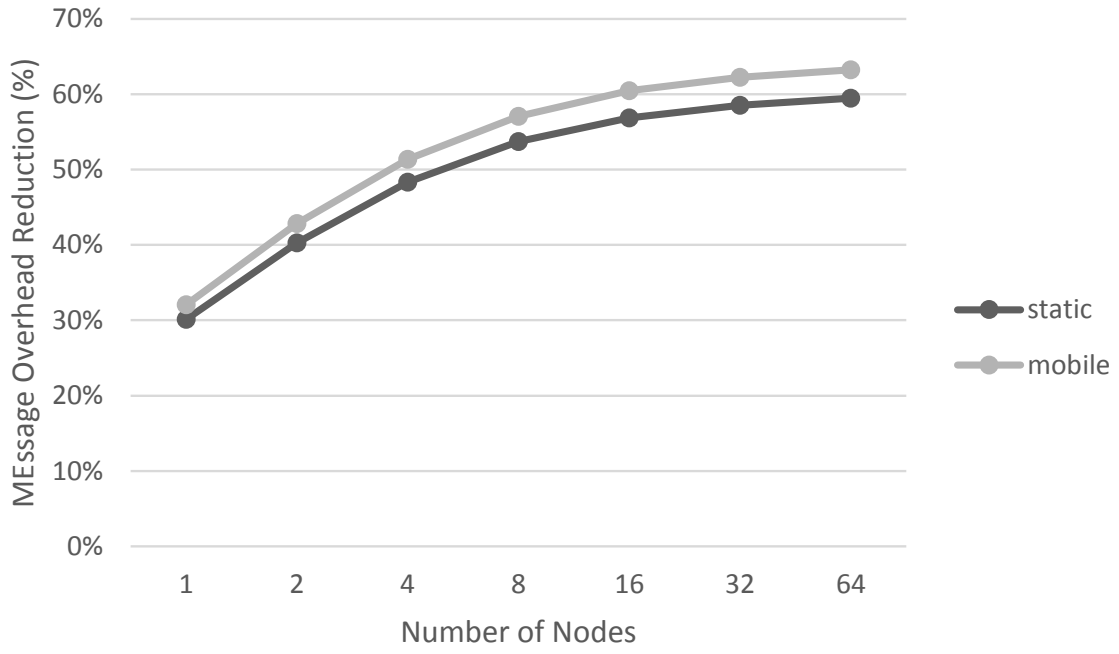


Figure 8. Overhead of messages of scenario 2

export some task from less capable devices to more powerful ones. It is possible to offload data traffic, applications, processing (among others) to any device or service available to handle such tasks. In this paper, we propose the offloading of routing decisions in ad hoc networks of mobile phones and tablets to servers.

Since with the increasing of the Internet infrastructure routing has become a costly solution for routers, several solutions have been proposed for wired networks. The Path Computation Element (PCE) [Farrel et al. 2006] was proposed in 2006, a solution that provides centralized constraint-based path computation for large, multi-domain networks. Following, solutions offloading routing to the virtual devices in clouds were proposed [Wei et al. 2008][Zhu et al. 2008][Karaoglu and Yuksel 2013].Finally, with the advent of Software Defined Networks (SDN) [Gupta et al. 2014] offloading solutions quickly began to appear taking advantage of the separation of control plane and data plane. For example. RouteFlow, uses OpenFlow-based SDN to provide routing services through a single controller.

The world of mobile networks is even more restricted since mobile devices are often not capable of handling some processes, because of their limited resources such as batteries, processing power, storage and bandwidth capacity. In order to save resources from restricted devices, several solutions have been proposed, not only specific for routing, but to several other applications [Li et al. 2001][Chen et al. 2004]. Following the evolution of wired networks solutions, SDN emerged in mobile scenarios with solutions that allow control plane functions to be separated from the devices.

In 2006, a offloading technique for H.264 video encoder was proposed [Zhao et al. 2006] because video processing applications are very resource consuming. They modularize the H.264 video encoder and offload some modules or the whole appli-

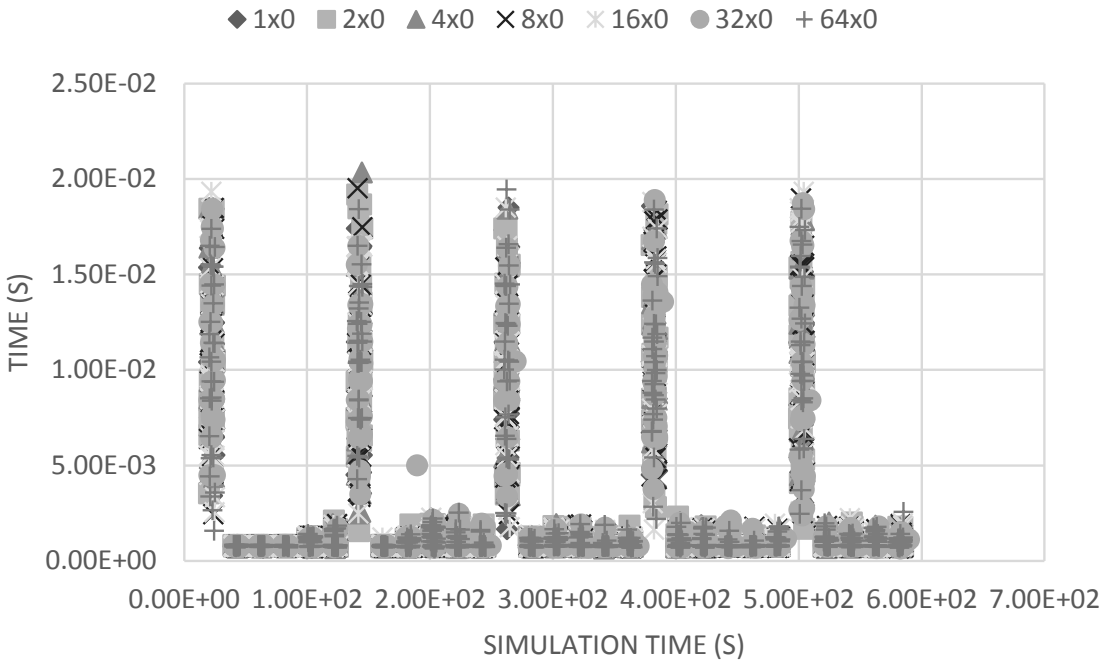


Figure 9. Routing delay of static scenario 2

cation to a nearby server. Results are shown in terms of energy saving, proving that nodes that offload video processing save energy. Later in 2009, the Stanford University released the OpenRoads [Yap et al. 2009], an open-source platform for wireless innovations based on OpenFlow. Specifically for offloading routing there is a solution that uses OpenFlow in wireless mesh networks [Dely et al. 2011], this paper proposes a centralized solution to provide routing decisions for all mesh nodes, the controller manages several networking functions, including handling mobility (handover).

Using the concepts of Offloading we propose DRR, a distributed strategy for offloading routing decisions of specific restricted devices, the HTR-lite nodes, to other devices. To the best of our knowledge, all the previously proposed solutions differ from ours, since ours is focused on distributed support to routing. Others, more capable, devices participating on the network assume routing decisions for the HTR-lite nodes.

7. Conclusion and Future Work

In this paper, we have presented a solution to the problem of routing table computation in heterogeneous ad hoc networks composed of special devices with limited resources. Our solution is based on the ability to offload the task of routing computation from restricted devices to more powerful ones.

The solution is successful in saving the resources such as processing power and network bandwidth. Even though we did not include a specific metric for processing power, since the simulator used does not provide the feature, we are offloading the routing table computation, which is a very demanding task. The experiments proved the bandwidth save at the special restricted nodes and the low routing delay resulting of the offloading process. Overall, the results of our evaluations were positive and confirm the

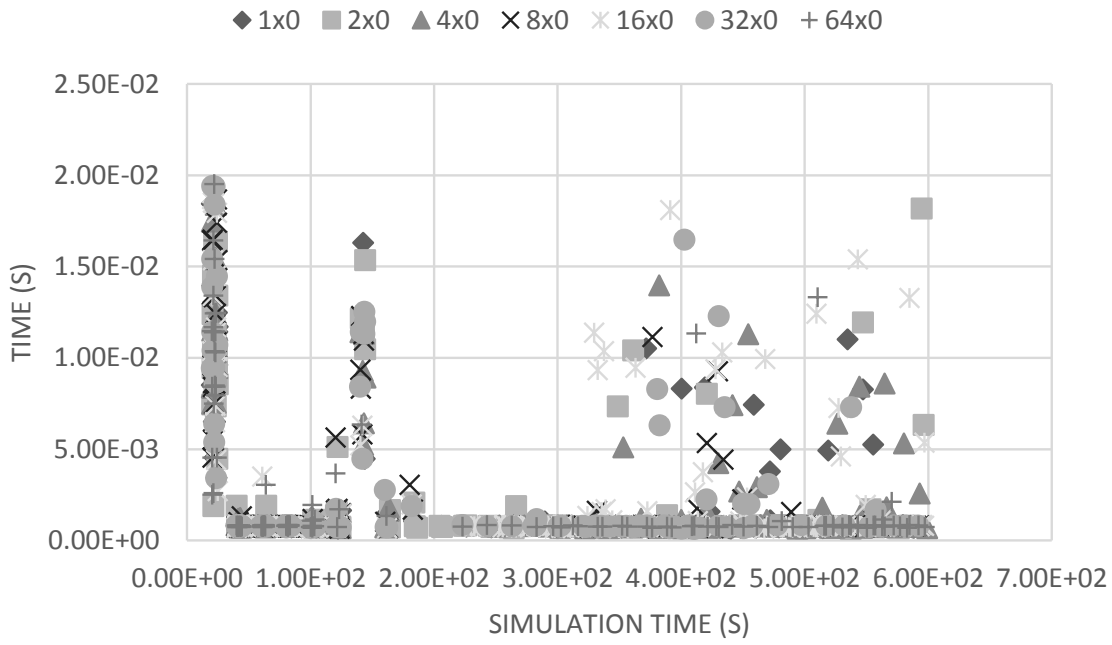


Figure 10. Routing delay of mobile scenario 2

potential applicability of the protocol.

We have also identified some limitations in our works, which are working to improve for a next version of the protocol. Currently, all non-restricted HTR nodes function as outsourced routers, which may not be the desired case. A centralized remote routing entity helped by proxies could be a better solution in certain scenarios.

We believe that by providing a routing offloading mechanism for ad hoc networks we may be doing a important step towards the definition of a software defined ad hoc network.

References

Chen, G., Kang, B.-T., Kandemir, M., Vijaykrishnan, N., Irwin, M., and Chandramouli, R. (2004). Studying energy trade offs in offloading computation/compilation in java-enabled mobile devices. *Parallel and Distributed Systems, IEEE Transactions on*, 15(9):795–809.

Clausen, T. and Jacquet, P. (2003). Rfc 3626. *Optimized link state routing protocol (OLSR)*.

Dely, P., Kassler, A., and Bayer, N. (2011). Openflow for wireless mesh networks. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–6. IEEE.

Farrel, A., Vasseur, J.-P., and Ash, J. (2006). A Path Computation Element (PCE)-Based Architecture. RFC 4655 (Informational).

Gupta, A., Vanbever, L., Shahbaz, M., Donovan, S., Schlinker, B., Feamster, N., Rexford, J., Shenker, S., Clark, R., and Katz-Bassett, E. (2014). Sdx: A software defined internet exchange. *Proceedings of the ACM SIGCOMM 2014 conference*. To Appear.

Karaoglu, H. and Yuksel, M. (2013). Offloading routing complexity to the cloud(s). In *Communications Workshops (ICC), 2013 IEEE International Conference on*, pages 1367–1371.

Li, Z., Wang, C., and Xu, R. (2001). Computation offloading to save energy on handheld devices: A partition scheme. In *Proceedings of the 2001 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, CASES '01, pages 238–246, New York, NY, USA. ACM.

Pei, G., Gerla, M., and Hong, X. (2000). Lanmar: Landmark routing for large scale wireless ad hoc networks with group mobility. In *Proceedings of the 1st ACM International Symposium on Mobile Ad Hoc Networking & Computing*, MobiHoc '00, pages 11–18, Piscataway, NJ, USA. IEEE Press.

Souto, E., Aschoff, R., Lima Junior, J., Melo, R., Sadok, D., and Kelner, J. (2012). Htr: A framework for interconnecting wireless heterogeneous devices. In *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, pages 645–649.

Wei, Y., Wang, J., and Wang, C. (2008). Bandwidth guaranteed multi-path routing as a service over a virtual network. In *Intelligent Networks and Intelligent Systems, 2008. ICINIS '08. First International Conference on*, pages 221–224.

Yap, K.-K., Kobayashi, M., Underhill, D., Seetharaman, S., Kazemian, P., and McKeown, N. (2009). The Stanford OpenRoads Deployment. pages 59 – 66, Beijing, China.

Younis, M., Youssef, M., and Arisha, K. (2002). Energy-aware routing in cluster-based sensor networks. In *Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 2002. MASCOTS 2002. Proceedings. 10th IEEE International Symposium on*, pages 129–136.

Zhao, X., Tao, P., Yang, S., and Kong, F. (2006). Computation offloading for h.264 video encoder on mobile devices. In *Computational Engineering in Systems Applications, IMACS Multiconference on*, volume 2, pages 1426–1430.

Zhu, Y., Zhang-Shen, R., Rangarajan, S., and Rexford, J. (2008). Cabernet: Connectivity architecture for better network services. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, pages 64:1–64:6, New York, NY, USA. ACM.

Eficiência dos Caminhos Quase Mais Curtos em Redes Dinâmicas

Dianne S. V. Medeiros[1], Miguel Elias M. Campista[1], Marcelo Dias de Amorim[2], Nathalie Mitton[3], Guy Pujolle[2]

[1]GTA / PEE / COPPE – UFRJ – Rio de Janeiro / RJ – Brasil
[2]LIP6 / CNRS – UPMC – Paris – França
[3]FUN – INRIA Lille – Ville Neuve d'Ascq – França

{dianne,miguel}@gta.ufrj.br, {marcelo.amorim,guy.pujolle}@lip6.fr,
nathalie.mitton@inria.fr

Abstract. *Betweenness centrality metrics often underestimate the importance of nodes close to the shortest path, but that rarely participate in them. In dynamic networks, these nodes can be, for a moment, in topologically strategic positions. This paper evaluates the importance of such nodes reusing the idea of spread betweenness centrality. This metric considers, besides the shortest paths, the multiple "quasi-shortest paths", assigning them a proportional weight. The impact of the metric on the network is evaluated through comparisons with other betweenness metrics. Results show that using the idea of spreadness can reclassify nodes, reducing the number of articulation points in the network among the most well classified nodes. Considering failure on the most central nodes, the throughput of the network generally remains higher when the spreadness is applied. This property can help to choose better the role of nodes in the network, such that the performance of networks with temporal dynamics is improved.*

Resumo. *Métricas de centralidade de intermediação frequentemente subestimam a importância dos nós próximos do caminho mais curto, mas que raramente participam deles. Em redes dinâmicas, esses nós podem se encontrar momentaneamente em posições topologicamente estratégicas. Este artigo avalia a importância desses nós reutilizando a ideia de centralidade de intermediação por espalhamento. Essa métrica considera, além dos caminhos mais curtos, os múltiplos "caminhos quase mais curtos", atribuindo-lhes um peso proporcional. O impacto da métrica na rede é avaliado através de comparações com outras métricas de intermediação. Os resultados mostram que o uso da ideia de espalhamento pode reclassificar nós, reduzindo o número de pontos de articulação na rede que estão dentre os nós mais bem classificados. Considerando falha nos nós mais centrais, a vazão da rede em geral mantém-se mais elevada quando o espalhamento é aplicado. Essa propriedade pode ajudar a escolher melhor o papel executado pelos nós de forma a melhorar o desempenho de redes com dinâmica temporal.*

1. Introdução

A importância de um nó para a rede pode ser quantificada utilizando-se métricas de centralidade da teoria de grafos, que geralmente associam a importância do nó à sua

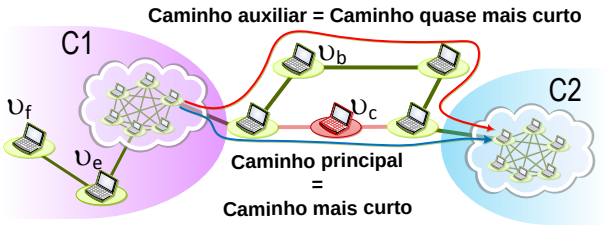


Figura 1. Exemplo de topologia de rede na qual as métricas típicas de centralidade de intermediação podem falhar quando tentam capturar a importância de nós críticos que participam mais de caminhos quase mais curtos. As nuvens representam qualquer tipo de topologia de rede conectada.

posição topológica na rede [Freeman 1978, Hui et al. 2008, Wehmuth and Ziviani 2013]. Os nós mais centrais em uma rede de computadores podem ser utilizados, por exemplo, para exercer funções de controle ou para disseminação de conteúdo [Kiss and Bichler 2008, Thilakarathna et al. 2013, Bouet et al. 2015]. Uma métrica de centralidade comumente utilizada é a intermediação, que atribui importância a um nó conforme o número de caminhos dos quais ele participa [Freeman 1978]. Dentre as métricas de intermediação existentes, uma das mais populares é a intermediação tradicional, que considera apenas a participação em caminhos mais curtos, e que pode beneficiar diversos protocolos de rede [Dolev et al. 2010, Giles et al. 2015, Yim et al. 2016, Jain 2016]. No entanto, o uso apenas desses caminhos como medida de importância dos nós pode acabar acarretando em um desperdício de potencial. Os nós em caminhos alternativos próximos aos caminhos mais curtos são subestimados e, consequentemente, impedidos de exercer um papel de maior importância para a rede. Na prática, esses nós podem ser bons candidatos para a manutenção da conectividade da rede em caso de falha de um nó mais central. A dúvida quanto à utilização apenas de caminhos mais curtos para quantificar a importância de um nó já foi debatida em diversos trabalhos na literatura [Freeman et al. 1991, Brandes and Fleischer 2005, Newman 2005, Borgatti and Everett 2006, Jiang et al. 2009, Opsahl et al. 2010, Medeiros et al. 2016a].

Neste artigo analisa-se uma nova versão da métrica de intermediação por espalhamento [Medeiros et al. 2016a], doravante *intermediação por espalhamento múltiplo*. De forma similar à métrica anterior, a nova versão considera os caminhos quase mais curtos dos quais um nó participa para quantificar a sua importância para a rede. A Figura 1 representa a ideia: υ_b se beneficia da utilização desses caminhos, aumentando sua importância em relação a υ_e, mas mantendo sua distância de υ_c que é o mais central dentre os nós destacados. Caso υ_c falhe, υ_b assume seu papel e o tráfego entre os dois componentes de rede `C1` e `C2` passa a ser desviado pelo caminho anteriormente quase mais curto. Em sua versão anterior [Medeiros et al. 2016a], a métrica era capaz de elevar a importância de nós que pertenciam a poucos ou nenhum caminho mais curto. No entanto, nós que participavam de um número pequeno de caminhos quase mais curtos eram frequentemente superestimados, tornando-se mais central do que outros que participavam de muitos caminhos mais curtos. Isso ocorria porque apenas os menores caminhos quase mais curtos entre dois nós eram considerados, tornando seus pesos excessivos. Para contornar esse problema, a nova versão proposta neste trabalho extrapola a centralidade de intermediação ponderada para considerar também os múltiplos caminhos quase mais cur-

tos encontrados dentro do limite estabelecido pelo espalhamento ρ. Em suma, essa nova versão utiliza a proporção de caminhos mais curtos e quase mais curtos dos quais um nó intermediário participa e pondera essa proporção por uma relação entre os custos dos caminhos. A adição dos múltiplos caminhos encontrados permite alterar a importância do nó intermediário ao mesmo tempo em que atribui maior importância para caminhos apenas um pouco mais longos do que o mais curto. O impacto da nova intermediação por espalhamento no ranqueamento dos nós está analisado em [Medeiros et al. 2016b].

A investigação conduzida neste trabalho é feita de forma comparativa, utilizando a centralidade de intermediação tradicional e a escalonada por distância. As métricas são aplicadas a uma rede móvel dinâmica, juntamente com a intermediação por espalhamento múltiplo. Primeiramente, a habilidade de reclassificação da intermediação por espalhamento múltiplo é analisada. Em seguida, a métrica é utilizada para estudar a conectividade de uma rede dinâmica em presença de falha nos nós mais centrais da rede. Para tanto, investiga-se os pontos de articulação que se encontram dentre os nós mais centrais e qual é o impacto provocado na vazão da rede caso ocorra uma falha em um desses nós centrais. Os resultados mostram que, (i) utilizando a nova versão da intermediação por espalhamento, um pequeno valor para o espalhamento ρ continua sendo suficiente para capturar a ideia da necessidade de utilização dos caminhos quase mais curtos, enquanto mantém-se a carga computacional baixa, sendo capaz de apontar nós cuja importância deveria ser reavaliada. Além disso, (ii) o número de pontos de articulação existentes dentre as posições classificatórias é sempre menor ou igual para a métrica estendida quando comparada à intermediação tradicional. Isso significa que menos pontos críticos existem dentre os nós mais centrais da rede. Por fim, de forma geral, (iii) a vazão da rede sofre menor redução quando ocorre falha nos nós mais bem classificados pela nova versão da intermediação por espalhamento. De forma resumida, as contribuições deste trabalho são:

- a identificação da necessidade de considerar também os múltiplos caminhos quase mais curtos existentes, além do menor deles, para determinar a importância de um nó para a rede;
- a proposta de uma nova versão da intermediação por espalhamento [Medeiros et al. 2016a] que leva em conta os múltiplos caminhos, chamada de intermediação por espalhamento múltiplo;
- a análise da conectividade de uma rede móvel dinâmica através de comparações entre a intermediação por espalhamento múltiplo e as intermediações tradicional e escalonada por distância.

Este artigo está organizado como segue. Na Seção 2 são introduzidas as definições e a notação utilizadas. A Seção 3 contextualiza este trabalho. A Seção 4 apresenta e formaliza a nova métrica de centralidade de intermediação por espalhamento. A avaliação da métrica reformulada e a sua aplicação ao estudo de conectividade são discutidas na Seção 5. Por fim, a Seção 6 conclui este artigo e apresenta os trabalhos futuros.

2. Notação e Definições

Uma rede pode ser modelada como um grafo ponderado $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \omega)$, em que $\mathcal{V}$ e $\mathcal{E}$ são os conjuntos de nós e enlaces, respectivamente, e o peso ω representa o custo de um enlace. Um nó v_i está conectado ao seu vizinho v_j através de um enlace $\varepsilon_{i,j}$ de custo

$\omega_{i,j} \in \mathbb{R}^*_+$. O enlace $\varepsilon_{j,i}$ existe, também, se a rede for não direcionada ou se v_j também está conectado ao vizinho v_i. Neste artigo, apenas grafos não direcionados são utilizados.

Um *caminho* $p_{i,j}$ entre uma origem v_i e um destino v_j é uma sequência ordenada de nós distintos na qual qualquer par consecutivo de nós está conectado por um enlace, não podendo conter laços. O *custo total*, $\delta_{i,j}$, do caminho $p_{i,j}$ é dado pela soma dos custos dos enlaces adjacentes $\varepsilon_{x,y}$ entre v_i e v_j. Neste trabalho utiliza-se pesos unitários para os enlaces, sem perda de generalidade, a fim de possibilitar o uso de algoritmos de menor complexidade temporal. Assim, $\delta_{i,j} \in \mathbb{N}^*$, sendo igual ao número de saltos entre v_i e v_j. O *caminho mais curto* $p^*_{i,j}$ apresenta o menor custo total $\delta^*_{i,j}$ e o número de caminhos mais curtos existentes entre um par de nós é dado por $n^*_{i,j}$. O total desses caminhos que passa por um intermediário v_k é representado por $n^*_{i,j}(v_k)$.

Os conceitos de *caminho quase mais curto* e *espalhamento* já foram introduzidos na literatura [Medeiros et al. 2016a]. Em suma, o espalhamento ρ é a diferença máxima tolerável entre os custos de um caminho qualquer e do caminho mais curto, ou seja, $\rho = \delta_{i,j} - \delta^*_{i,j}$. Assim, um caminho é dito quase mais curto se $\delta_{i,j} - \delta^*_{i,j} \leq \rho$. O espalhamento limita a profundidade de busca por caminhos, evitando a explosão do número de possibilidades, e excluindo caminhos muito mais longos do que o mais curto. Essa consideração surge do fato de que a vazão da informação se concentra em caminhos de custo não muito superior ao do caminho mais curto. O número de caminhos quase mais curtos entre dois nós é dado por $n_{i,j}$ e o total desses caminhos que passa por um intermediário v_k é dado por $n_{i,j}(v_k)$. Esses caminhos podem aumentar a importância de nós ignorados ou subestimados quando apenas os caminhos mais curtos são considerados.

Modelar redes como grafos permite analisar propriedades interessantes para redes de computadores, como a conectividade. Uma rede é *conexa* se existe pelo menos um caminho entre todos os pares de nós, e é *biconexa* se existem pelo menos dois caminhos disjuntos entre eles. O estudo da conectividade permite identificar nós críticos para a rede, que ao falharem podem desconectá-la ou aumentar o número de componentes conexos. Quando isso ocorre, esses nós são ditos *pontos de articulação*. Formalmente, v_a é um ponto de articulação se existirem dois vértices $v_i, v_j \in \mathcal{V}$, com $v_i \neq v_j \neq v_a$, tal que todos os caminhos $p_{i,j}$ contêm v_a, tenham eles custo $\delta_{i,j}$ ou $\delta^*_{i,j}$. A presença desses pontos na rede pode ser identificada determinando-se se ela é biconexa. Caso negativo, existe pelo menos um ponto de articulação. Esses pontos constituem uma grave vulnerabilidade, uma vez que uma única falha em um deles pode dividir a rede em vários componentes conexos.

3. Centralidade, Intermediação e Pontos de Articulação

O estudo da conectividade da rede é de fundamental importância para o projeto de redes tolerantes a falhas. O principal objetivo desse estudo é identificar os pontos críticos da rede para que, ou eles possam ser protegidos, ou passem a não ser mais críticos através de modificações na topologia. Uma possível forma de protegê-los é identificar todos eles, o que pode ser uma tarefa não trivial [Zhang and Sterbenz 2014]. Intuitivamente, espera-se que as métricas de centralidade tenham algum tipo de relação com a identificação desses pontos [Ausiello et al. 2013], uma vez que elas são bastante utilizadas para quantificar a importância de um nó para a rede. De fato, apesar de não identificar todos eles, a literatura confirma que, em geral, pontos de articulação apresentam valores de intermediação muito maiores do que nós ordinários [Ausiello et al. 2013]. Alternati-

vamente, em caso de um ataque externo, a não identificação desses nós de certa forma já é uma medida de proteção. Por exemplo, quando os nós mais centrais da rede exercem alguma função de controle, o ideal é que nenhum deles seja um ponto de articulação, uma vez que esses nós seriam visados por atacantes. Assim, se a localização das funções é feita utilizando alguma métrica de centralidade, quanto menos pontos de articulação estiverem dentre os nós mais centrais da rede, maior será a resistência dessa rede a falhas.

Alguns exemplos de centralidade são o grau, a proximidade e a intermediação [Freeman 1978, Hui et al. 2008, Opsahl et al. 2010, Wehmuth and Ziviani 2013]. Enquanto o grau se relaciona com a popularidade de um nó, a proximidade diz respeito à velocidade com que um nó consegue espalhar ou acessar recursos, por exemplo, informação. Este trabalho foca na intermediação, que está relacionada ao controle que um nó pode exercer sobre os fluxos entre outros nós da rede [Freeman 1978]. Essa métrica foi introduzida por Freeman [Freeman 1977], com base em intuições reveladas em trabalhos anteriores que utilizavam caminhos para determinar a importância de um nó [Bavelas 1948, Shimbel 1953, Shaw 1954, Cohn and Marriott 1958].

A ideia da *centralidade de intermediação tradicional* é a de que quanto mais um nó υ_k participa de caminhos mais curtos entre outros nós, mais central ele é. Consequentemente, esses nós estão posicionados de forma estratégica para influenciar a rede através do controle do fluxo de informação entre pares. Freeman assume que uma mensagem passa por um dos caminhos mais curtos entre dois nós com probabilidade igual a $1/n^*_{i,j}$ [Freeman 1977]. Considerando todos os pares de nós na rede, a chance de υ_k fazer parte de um dos caminhos mais curtos escolhido aleatoriamente entre um par qualquer de nós define matematicamente a intermediação tradicional de υ_k, $B_{trad}(\upsilon_k)$, como a seguir.

$$B_{trad}(\upsilon_k) = \sum_{i \in |\mathcal{V}|} \sum_{j \in |\mathcal{V}|} \frac{n^*_{i,j}(\upsilon_k)}{n^*_{i,j}}, \text{ onde } i \neq k,\ j \neq i \text{ e } j \neq k.$$

A intermediação tradicional limita-se a grafos simples e não considera o peso dos enlaces, ignorando, assim, o custo fim-a-fim do caminho entre dois nós. Esse custo é importante porque caminhos mais longos são menos valiosos para serem controlados ou podem não ser realistas em algumas redes [Borgatti and Everett 2006]. Além disso, considerar também que o fluxo da informação sempre passa por caminhos mais curtos pode não ser verdade em alguns casos. Por exemplo, rumores se espalham de forma aleatória ou podem ser canalizados intencionalmente por intermediários específicos [Stephenson and Zelen 1989]. Por essa razão, diversos trabalhos já questionaram o uso apenas de caminhos mais curtos para definir a importância de um nó [Freeman et al. 1991, Brandes and Erlebach 2005, Brandes and Fleischer 2005, Newman 2005, Borgatti and Everett 2006, Geisberger et al. 2008, Jiang et al. 2009, Opsahl et al. 2010, Medeiros et al. 2016a].

A limitação da intermediação tradicional em relação ao comprimento do caminho fim-a-fim é solucionada pela *centralidade de intermediação escalonada por distância* [Borgatti and Everett 2006] (*Distance Scaled Betweenness* – DS), que pondera a frequência com que um nó participa dos caminhos mais curtos utilizando com o inverso do comprimento do caminho . A métrica DS é definida a seguir.

$$B_{dist}(\upsilon_k) = \sum_{i \in |\mathcal{V}|} \sum_{j \in |\mathcal{V}|} \left(\frac{1}{\Delta L^*_{i,j}} \times \frac{n^*_{i,j}(\upsilon_k)}{n^*_{i,j}} \right).$$

4. Centralidade de Intermediação por Espalhamento Múltiplo

Este trabalho propõe uma nova versão da centralidade de intermediação por espalhamento [Medeiros et al. 2016a] para considerar os múltiplos caminhos existentes entre dois nós quaisquer da rede, e não apenas os caminhos mais curtos e os menores caminhos quase mais curtos. À versão estendida dá-se o nome de intermediação por espalhamento múltiplo. A inclusão desses caminhos adicionais permite capturar melhor o potencial dos nós intermediários que podem ser críticos para a rede, refinando a contribuição de cada caminho individual. Esses nós costumam ser desprezados pelas métricas de intermediação baseadas em caminhos mais curtos por não participarem de um número suficientemente grande de caminhos mais curtos. Por outro lado, alguns desses nós podem ser superestimados ao se considerar apenas os mais curtos dentre os caminhos quase mais curtos [Medeiros et al. 2016a]. Esse efeito se repete quando não são atribuídos pesos adequados aos caminhos, como é o caso da k-intermediação [Jiang et al. 2009], que não pondera os caminhos de acordo com o seu custo.

A Figura 1 ilustra o problema dos nós subestimados pelas métricas de intermediação típicas. Os nós v_c e v_b são cruciais para a manutenção da conexão entre os componentes de rede `C1` e `C2`, enquanto v_e conecta um nó da borda ao restante da rede. Segundo a intermediação tradicional, v_c possui maior centralidade dentre os três nós analisados, seguido por v_e e finalmente por v_b. Essa métrica desconsidera o potencial de v_b de assumir um papel muito mais importante do que o de v_e considerando-se a conectividade da rede inteira, principalmente em caso de falha do nó v_c. Nós em posições semelhantes à posição topológica de v_b são bons candidatos para agir como nós substitutos de outros nós mais importantes, caso esses venham a falhar, mantendo a rede conectada a um custo apenas um pouco mais elevado. O grau de importância dos nós v_b, v_c e v_e para a rede é reforçado pela centralidade de intermediação escalonada por distância. No entanto, ao considerar caminhos quase mais curtos, é possível modificar essa classificação, aumentando a importância de v_b, que se torna mais importante do que v_e (Tabela 1).

Um segundo problema encontrado ao analisar redes utilizando métricas de intermediação baseadas apenas em caminhos mais curtos é a classificação como iguais de nós topologicamente diferentes. A Figura 2 ilustra esse problema. Considerando-se o conjunto de nós $\{v_a, v_b, v_c, v_d, v_e\}$, está claro que os nós v_a e v_e são topologicamente semelhantes, assim como v_b e v_d. Apesar do nó v_c ser diferente dos outros, tanto a intermediação tradicional quanto a escalonada por distância (DS) o classificam como igual aos nós v_a e v_e. Ao se considerar os múltiplos caminhos quase mais curtos, v_c passa a ser diferenciado, como visto na Tabela 2. Isso ocorre porque nenhum caminho mais curto passa por v_c, mas os caminhos quase mais curtos existentes entre quaisquer pares de nós na rede sempre têm v_c como intermediário.

Tabela 1. Comparação entre as classificações dos nós destacados na Figura 1 utilizando as métricas de intermediação tradicional (Trad), escalonada por distância (DS) e por espalhamento múltiplo (B_ρ).

Node	Trad	DS	B_ρ ($\rho = 3$)
v_c	63,0	12,1	117,10
v_b	9,0	2,5	39,56
v_e	17,0	3,9	35,62

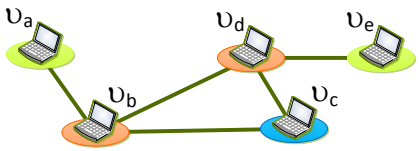


Figura 2. De acordo com sua posição topológica, os nós v_a e v_e são iguais, assim como v_b e v_d. O nó v_c é diferente de todos os outros, mas tanto a intermediação tradicional quanto a escalonada por distância consideram que v_c está posicionado de forma semelhante a v_a e v_e.

4.1. Formalização

Assim como a centralidade de intermediação por espalhamento [Medeiros et al. 2016a], o custo máximo do caminho quase mais curto considerado depende de ρ. Se $\rho = C$, com $C \in \mathbb{N}$, uma vez que neste trabalho os custos são unitários para permitir o uso de algoritmos de menor complexidade temporal, o custo máximo considerado para esses caminhos será dado por $\delta^*_{i,j} + C$. Assim, um caminho de custo $\delta^*_{i,j} + (C + \varphi)$, com $\varphi \in \mathbb{N}^*$, será ignorado na computação da métrica. Contudo, diferentemente da intermediação por espalhamento, que considerava apenas os menores caminhos quase mais curtos, neste trabalho estende-se a métrica para incluir todos os caminhos com custo $\delta_{i,j} \leq \delta^*_{i,j} + C$. A métrica estendida considera tanto o custo dos caminhos quanto a ideia de utilizar outros caminhos além dos mais curtos, além de supor que a informação conhece seu destino final e *prefere* trafegar por caminhos menos longos para chegar até ele. Para tal, a centralidade de intermediação por espalhamento múltiplo pondera a contribuição dos caminhos para a importância do nó intermediário v_k de forma proporcional aos custos desses caminhos, utilizando a razão entre os custos do caminho mais curto, $\delta^*_{i,j}$, e do caminho quase mais curto que passa por v_k, $\delta_{i,j} = \delta_{i,k} + \delta_{k,j}$. Assim, atribui-se maior importância para nós em caminhos de menor comprimento. A nova versão da métrica, chamada a partir deste trabalho de intermediação por espalhamento múltiplo, está formalizada na Equação 1. Nota-se que para $\rho = 0$, $\delta_{i,j} = \delta^*_{i,j}$, e apenas os caminhos mais curtos são considerados.

$$B_\rho(v_k) = \sum_{\substack{i \in |\mathcal{V}| \\ \delta_{i,k}+\delta_{k,j}-\delta^*_{i,j} \leq \rho}} \sum_{j \in |\mathcal{V}|} \left(\frac{n^*_{i,j}(v_k) + n_{i,j}(v_k)}{n^*_{i,j} + n_{i,j}} \times \frac{\delta^*_{i,j}}{\delta_{i,k} + \delta_{k,j}} \right). \quad (1)$$

4.2. Propriedades

A centralidade de intermediação por espalhamento múltiplo possui complexidade de tempo polinomial comparável a das métricas discutidas na Seção 3 e apresenta as

Tabela 2. Comparação entre as intermediações dos nós na Figura 2. Apenas a intermediação por espalhamento múltiplo captura a importância de v_c.

Node	Trad	DS	B_ρ ($\rho = 3$)
v_a	0,0	0,0	0,0
v_b	3,0	1,3	2,7
v_c	0,0	0,0	1,3
v_d	3,0	1,3	2,7
v_e	0,0	0,0	0,0

propriedades a seguir: (i) considera o número de múltiplos caminhos, tanto mais curtos quanto quase mais curtos; (ii) aumenta com a participação de v_k em ambos os caminhos mais curtos e quase mais curtos; (iii) prioriza caminhos menores diminuindo a contribuição de caminhos mais longos através de uma relação entre custos; e (iv) aumenta com a centralidade do nó. Note que, neste trabalho, um nó é mais central se ele participa de múltiplos caminhos, sejam eles mais curtos ou quase mais curtos. Essa consideração advém do fato de que nós que participam em vários caminhos quase mais curtos não devem ser descartados apenas por não estarem em caminhos mais curtos, uma vez que eles podem ganhar importância. Por exemplo, nós que estão sempre próximos ao caminho mais curto podem substituir outros nós mais importantes caso eles falhem.

5. Avaliação

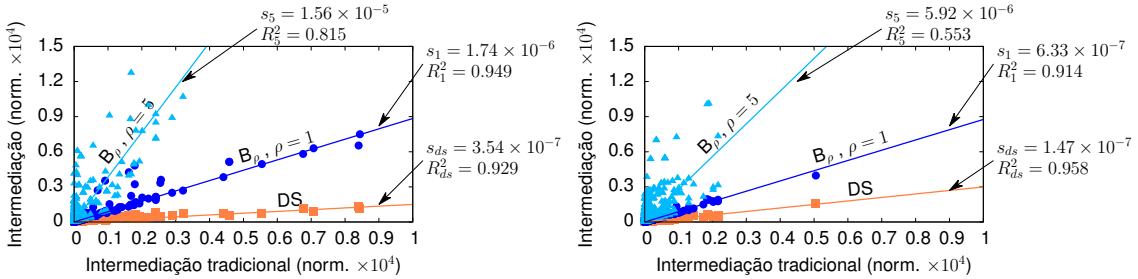
O impacto e a relevância da métrica proposta é avaliado utilizando-se um conjunto de dados dinâmico. A investigação é realizada para valores de espalhamento $\rho \leq 5$, ou seja, consideram-se os múltiplos caminhos quase mais curtos que obedecem a restrição imposta pela inequação $\delta_{i,j} \leq \delta^*_{i,j} + \{1, 2, 3, 4, 5\}$.

A avaliação da métrica é realizada através de comparações com as intermediações tradicional e escalonada por distância, aplicadas ao conjunto de dados TAPAS-Cologne [Uppoor and Fiore 2011], que modela o tráfego de veículos na cidade de Colônia, na Alemanha [Uppoor and Fiore 2011], durante 24 horas. No momento em que este trabalho foi elaborado, apenas um subconjunto de 2 horas estava disponível publicamente na Internet. Desse subconjunto analisa-se 6 amostras tomadas a cada 10 segundos. Cada um dos nós representa um veículo no cenário e um enlace existe entre dois nós se eles estiverem a menos de 50 metros de distância um do outro. O número de nós nas amostras varia entre 1.584 e 1.916 e o número de enlaces não direcionados, entre 1.573 e 2.044, dependendo da amostra. Note que a variação no raio de alcance rádio modifica a densidade do grafo da rede, aumentando ou diminuindo o número de enlaces e, consequentemente, o número de possíveis caminhos entre dois nós. Inicialmente, verifica-se quão próximas da intermediação tradicional estão as intermediações escalonada por distância (DS) e por espalhamento múltiplo. Supõe-se que devido ao modo como essas métricas são formalizadas, elas devem ser semelhantes à intermediação tradicional. Contudo, deseja-se que, mesmo sendo fortemente correlacionadas, essas métricas sejam capazes de apontar nós que deveriam ser reclassificados. Em seguida, investiga-se a conectividade da rede através do estudo dos pontos de articulação existentes na rede, classificados segundo as mesmas métricas anteriores. Por fim, através de simulação, avalia-se o desempenho da rede TAPASCologne quanto à vazão média alcançada em presença de falha única, considerando-se os nós mais importantes da rede de acordo com cada métrica.

Os resultados estão divididos como segue: o primeiro refere-se à habilidade de identificação de nós cuja importância deveria ser reavaliada. Já o segundo relaciona-se com o impacto das métricas no desempenho de uma rede dinâmica na presença de falhas.

5.1. Identificação de nós reclassificáveis

Investiga-se a relação entre a intermediação tradicional e as intermediações escalonada por distância e por espalhamento múltiplo, a fim de verificar como os requisitos adicionais de cada métrica influenciam a semelhança entre elas. Essa análise permite,



(a) Amostra #1. (b) Amostra #4.

Figura 3. As métricas analisadas são fortemente correlacionadas com a intermediação tradicional, sendo essa correlação mais forte para a intermediação escalonada por distância. A intermediação por espalhamento múltiplo consegue identificar mais nós que deveriam ser reavaliados.

também, averiguar se as métricas conseguem destacar nós que deveriam ser reclassificados. Os resultados obtidos para as amostras $\#1$ e $\#3$ do conjunto de dados são mostrados na Figura 3, onde o eixo x representa a intermediação tradicional normalizada e cada curva refere-se a uma das outras duas métricas, também normalizadas. Note que os eixos na Figura 3 estão escalonados para melhor visualização. Apenas as curvas referentes aos valores de espalhamento 1 e 5 são mostradas, para maior clareza. As curvas para os outros valores de ρ encontram-se entre as duas curvas exibidas.

A Figura 3 mostra que todas as métricas são fortemente correlacionadas com a intermediação tradicional, uma vez que o coeficiente de determinação (R^2) é elevado. A correlação é mais forte para a intermediação escalonada por distância, seguida pela por espalhamento múltiplo utilizando $\rho = 1$. A correlação diminui com o aumento do espalhamento, devido ao crescimento da participação dos nós em caminhos quase mais curtos. Isso faz com que mais nós sejam apontados pela métrica como sub ou superestimados, sugerindo a necessidade de reclassificação. Esse comportamento se repete em todas as amostras tomadas. Em todo caso, um espalhamento $\rho = 1$ já é suficiente para destacar alguns nós, o que pode ser notado pela dispersão dos pontos em torno das curvas ou, matematicamente, através da análise combinada do desvio padrão de cada ajuste de curva e do coeficiente R^2. Quanto maior o desvio padrão e menor o R^2, mais nós reclassificáveis podem ser identificados. No entanto, R^2 não pode ser menor do que $0,35$, uma vez que, assim sendo, a correlação entre as métricas passaria a ser moderada, o que significa que elas medem duas características diferentes, o que não é do interesse deste trabalho.

Note que a possibilidade de reclassificação dos nós surge neste cenário apenas porque existem caminhos alternativos entre dois nós na rede. Caso poucos caminhos múltiplos existam, a intermediação por espalhamento múltiplo não se beneficia da existência de muitos caminhos quase mais curtos, de forma que passa a ser praticamente igual à intermediação tradicional, independentemente do valor de espalhamento.

5.2. Impacto no desempenho da rede

A conectividade da rede é analisada neste trabalho através do estudo dos pontos de articulação e do número de componentes conexos existentes. A Figura 4 mostra o número de pontos de articulação presentes em cada amostra estudada do conjunto de dados TAPASCologne, considerando-se as 5 posições que contêm os nós mais bem classifi-

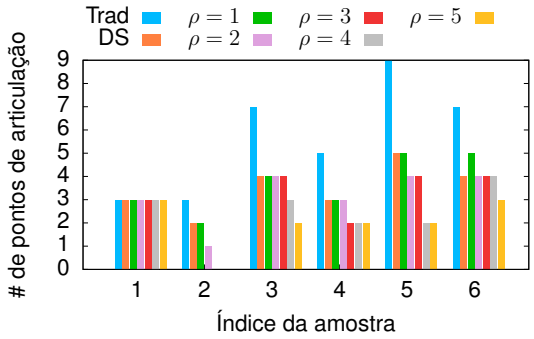


Figura 4. O número de pontos de articulação existentes em cada amostra do cenário dinâmico considerando-se as top $\#5$ posições é, em geral, menor para a intermediação escalonada por distância (DS) e para a por espalhamento múltiplo, quando comparadas à intermediação tradicional.

cados segundo cada métrica analisada, doravante denominadas top $\#5$ posições. Observa-se que o número de pontos de articulação dentre as top $\#5$ posições muda dependendo da amostra, mas fica claro que tanto a intermediação escalonada por distância (DS) quanto a por espalhamento múltiplo apresentam um número menor de pontos de articulação em posições de elevada importância. A Tabela 3 mostra o número de componentes existentes anteriormente a qualquer falha na rede para cada amostra estudada, e após uma falha única em um ponto de articulação escolhido dentre as top $\#5$ posições. Em todas as amostras, pelo menos um componente a mais é criado caso um dos pontos de articulação falhe, sendo que na amostra $\#4$ até dois componentes a mais podem ser criados, dependendo da posição classificatória do nó.

Nós que apresentam elevada intermediação são responsáveis por uma grande quantidade de fluxos entre pares de nós na rede. Dependendo da topologia dessa rede, uma falha que ocorra em um único desses nós pode afetar inúmeros fluxos, principalmente se esse nó também for um ponto de articulação. Isso pode reduzir drasticamente o desempenho da rede. A severidade dessas falhas é investigada utilizando-se o simulador de redes NS-3. O objetivo da simulação é analisar para cada métrica, em relação à intermediação tradicional, o comportamento da vazão média obtida para a rede antes e após a ocorrência de uma falha única em um nó que apresenta intermediação elevada. Considera-se que todos os nós têm a intenção de se comunicar com todos os outros nós da rede, mesmo que não estejam em um mesmo componente conectado. Contudo, se dois nós em componentes distintos tentarem se comunicar, não serão bem sucedidos, uma vez que não existe caminho entre eles. As mensagens trocadas devem possuir o mesmo tamanho para todos os pares de nós. Contudo, o tamanho específico escolhido não altera a conclusão. Supõe-se, também, que a topologia da rede muda a cada 10 segundos apenas,

Tabela 3. Número de componentes existentes em cada amostra do conjunto de dados TAPASCologne quando não existem falhas na rede e quando um único nó ponto de articulação falha.

	#1	**#2**	**#3**	**#4**	**#5**	**#6**
Sem falha	550	549	546	576	596	578
Falha única	551	550	547	577 / 578	597	579

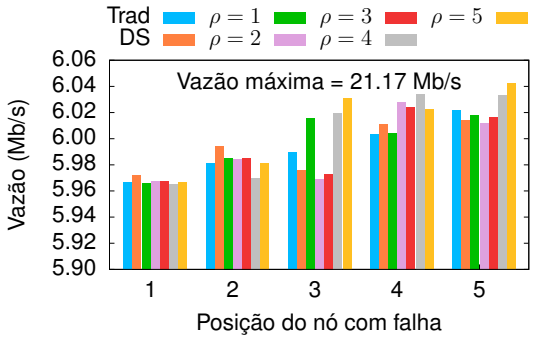


Figura 5. Em relação às intermediações tradicional e escalonada por distância (DS), a falha em nós classificados nas top $\#5$ posições da intermediação por espalhamento múltiplo é, em geral, menos prejudicial à vazão média da rede.

que é o tempo de intervalo entre as amostras tomadas do conjunto de dados TAPAS-Cologne. Por fim, o nó que sofre a falha é escolhido aleatoriamente dentro da posição classificatória investigada. Isto é, ao se investigar o impacto da falha de um nó na posição n, e existem m nós classificados nessa posição, escolhe-se aleatoriamente um desses m nós para falhar. Os resultados relatados nesta seção são dependentes do ranqueamento obtido utilizando cada métrica. Dessa forma, redes mais esparsas, que possuem menos alternativas de caminhos entre pares de nós, devem apresentar comportamento semelhante ao encontrado para a intermediação tradicional. Caso a oferta de caminhos alternativos seja maior, a diferença entre as métricas será mais acentuada.

A Figura 5 mostra o resultado obtido para a simulação utilizando-se a classificação gerada pelas intermediações tradicional, escalonada por distância e por espalhamento múltiplo. O eixo x representa a posição classificatória do nó no qual ocorreu a falha, e o eixo y é a vazão total média da rede, em Mb/s, calculada para os 60 segundos de simulação, referentes ao tempo total das amostras. Note que o nó selecionado para falhar pode ou não ser um ponto de articulação da rede. A probabilidade de o nó em falha ser também um ponto de articulação difere para cada métrica e está mostrada na Tabela 4. Nota-se que, para cada amostra, essa probabilidade é sempre igual ou maior para a intermediação tradicional do que para as outras métricas, sendo maior na maioria dos casos. Além disso, a probabilidade de o nó em falha também ser ponto de articulação diminui com o aumento do espalhamento ρ e, em geral, é maior para a intermediação escalonada por distância (DS) do que para a intermediação por espalhamento múltiplo.

Tabela 4. A probabilidade de um nó em falha também ser ponto de articulação é maior na maioria dos casos para intermediação tradicional do que para as outras métricas.

Amostra	**Trad**	**DS**	B_ρ ($\rho = 1$)	B_ρ ($\rho = 2$)	B_ρ ($\rho = 3$)	B_ρ ($\rho = 4$)	B_ρ ($\rho = 5$)
$\#1$	0,60	0,60	0,60	0,60	0,60	0,60	0,60
$\#2$	0,43	0,33	0,33	0,17	0,0	0,0	0,0
$\#3$	0,88	0,80	0,80	0,80	0,80	0,38	0,25
$\#4$	0,72	0,60	0,60	0,60	0,40	0,40	0,40
$\#5$	1,0	1,0	1,0	0,67	0,80	0,33	0,33
$\#6$	1,0	0,80	1,0	0,80	0,80	0,80	0,43

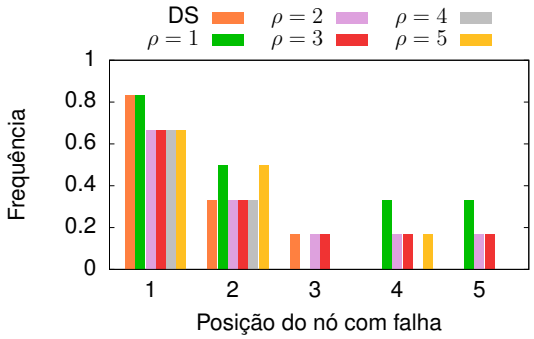


Figura 6. Os nós classificados na primeira posição frequentemente são os mesmos em relação ao ranqueamento da intermediação tradicional, considerando-se todas as amostras tomadas do conjunto de dados de TAPASCologne, enquanto nas posições seguintes essa frequência diminui consideravelmente.

Destaca-se na Figura 5 a vazão máxima alcançada, que ocorre na ausência de falhas na rede. Seu valor é aproximadamente igual a 21,17 Mb/s e é utilizado nesta análise como valor de referência. De uma forma geral, observa-se que quanto menor a importância do nó, isto é, quanto mais distante da primeira posição classificatória, menor é o impacto da falha na vazão média da rede. No entanto, essa variação é bem pequena, por exemplo, a ocorrência de falha no nó mais importante, que também é um ponto de articulação, reduz a vazão total em aproximadamente 28,18%. Já para a última posição, a redução varia entre 28,40% e 28,54%, dependendo da métrica analisada. Verifica-se ainda, que para a intermediação por espalhamento múltiplo, a perda na vazão total média é sempre muito próxima ou menor do que a perda em caso de falha de nós importantes segundo a classificação da intermediação tradicional. É importante notar que frequentemente os nós classificados como mais importantes geralmente são os mesmos e, por essa razão, a variação da vazão dentro de uma mesma posição classificatória é pequena. A Figura 6 mostra com que frequência durante os 60 segundos de simulação os nós nas top $\#5$ posições mudam. Por exemplo, em 83% do tempo o nó classificado na primeira posição é o mesmo tanto para a intermediação tradicional quanto para a escalonada por distância. Em relação à intermediação por espalhamento múltiplo, esse valor cai para 67% do tempo, mas ainda assim não é suficiente para fazer alguma diferença na vazão média da rede. Somente para um nível de coincidência menor do que 50% é que observa-se influência na vazão média da rede.

6. Conclusões

Este trabalho reutiliza a ideia da intermediação por espalhamento, estendendo-a para considerar os múltiplos caminhos quase mais curtos existentes entre pares de nós. O objetivo é melhorar a avaliação da importância dos nós para a rede, mesmo quando eles não participam de caminhos mais curtos. Esses nós podem ser críticos para a operação da rede, reduzindo a reorganização e os custos envolvidos após falha de um nó central. O impacto da métrica foi avaliado através de comparações com outras métricas de intermediação, aplicadas a um cenário de rede dinâmico. Os resultados mostraram que a nova versão da intermediação por espalhamento é capaz de apontar nós que deveriam ser reavaliados. Ao reclassificá-los, é possível reduzir o número de pontos de articulação em posições centrais na rede, considerando-se os nós mais bem classifica-

dos. Em caso de falha nesses nós, simulações mostraram que, em geral, seguindo o ranqueamento fornecido pela intermediação por espalhamento múltiplo, a vazão média total da rede é menos prejudicada do que quando se utiliza as outras métricas. A probabilidade do nó em falha também ser um ponto de articulação é frequentemente menor para a intermediação por espalhamento múltiplo e, comparada à intermediação tradicional, em geral os nós classificados na primeira posição são os mesmos, mas para as posições seguintes eles costumam ser diferentes. Como trabalhos futuros, planeja-se estender o algoritmo da nova métrica para atender redes ponderadas e comparar o impacto no desempenho da rede quando ataques ocorrerem apenas nos pontos de articulação e apenas nos nós mais centrais que não sejam críticos. Também planeja-se aplicar a métrica a redes reais através de experimentos em plataformas realísticas, e estudar a sua relevância em diferentes casos de uso.

Agradecimento

Os autores agradecem ao CNPq, à CAPES e à Faperj pelo financiamento parcial deste trabalho.

Referências

Ausiello, G., Firmani, D., e Laura, L. (2013). The (betweenness) centrality of critical nodes and network cores. Em *IWCMC*, pp. 90–95.

Bavelas, A. (1948). A mathematical model for group structures. *Human Organization*, 7(3):16–30.

Borgatti, S. P. e Everett, M. G. (2006). A graph-theoretic perspective on centrality. *Social Networks*, 28(4):466 – 484.

Bouet, M., Leguay, J., Combe, T., e Conan, V. (2015). Cost-based placement of vDPI functions in NFV infrastructures. *International Journal of Network Management*, 25(6):490–506.

Brandes, U. e Erlebach, T. (2005). *Network Analysis: Methodological Foundations*. Springer, 1a Edição.

Brandes, U. e Fleischer, D. (2005). Centrality measures based on current flow. Em *STACS*, pp. 533–544.

Cohn, B. S. e Marriott, M. (1958). Networks and centres of integration in Indian civilization. *Journal of Social Research*, 1:1–9.

Dolev, S., Elovici, Y., e Puzis, R. (2010). Routing betweenness centrality. *JACM*, 57(4):25:1–25:27.

Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41.

Freeman, L. C. (1978). Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239.

Freeman, L. C., Borgatti, S. P., e White, D. R. (1991). Centrality in valued graphs: A measure of betweenness based on network flow. *Social Networks*, 13(2):141–154.

Geisberger, R., Sanders, P., e Schultes, D. (2008). Better approximation of betweenness centrality. Em *ALENEX*, pp. 90–100.

Giles, A. P., Georgiou, O., e Dettmann, C. P. (2015). Betweenness centrality in dense random geometric networks. Em *ICC*, pp. 6450–6455.

Hui, P., Crowcroft, J., e Yoneki, E. (2008). Bubble rap: Social-based forwarding in delay tolerant networks. Em *ACM Mobihoc*, pp. 241–250.

Jain, A. (2016). Betweenness centrality based connectivity aware routing algorithm for prolonging network lifetime in wireless sensor networks. *Wireless Networks*, 22(5):1605–1624.

Jiang, K., Ediger, D., e Bader, D. A. (2009). Generalizing k-betweenness centrality using short paths and a parallel multithreaded implementation. Em *ICPP*, pp. 542–549.

Kiss, C. e Bichler, M. (2008). Identification of influencers - measuring influence in customer networks. *Decision Support Systems*, 46(1):233–253.

Medeiros, D. S. V., Campista, M. E. M., Mitton, N., de Amorim, M. D., e Pujolle, G. (2016a). Intermediação por espalhamento: Caminhos quase mais curtos também importam. Em *SBRC*, pp. 967–980.

Medeiros, D. S. V., Campista, M. E. M., Mitton, N., de Amorim, M. D., e Pujolle, G. (2016b). Spread betweenness centrality: The power of quasi-shortest paths. Relatório técnico, GTA – PEE / COPPE / UFRJ. http://www.gta.ufrj.br/ftp/gta/TechReports/MCM16c.pdf.

Newman, M. J. (2005). A measure of betweenness centrality based on random walks. *Social Networks*, 27(1):39 – 54.

Opsahl, T., Agneessens, F., e Skvoretz, J. (2010). Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32(3):245–251.

Shaw, M. E. (1954). Group structure and the behavior of individuals in small groups. *The Journal of Psychology*, 38(1):139–149.

Shimbel, A. (1953). Structural parameters of communication networks. *Bulletin of Mathematical Biophysics*, 15(4):501–507.

Stephenson, K. e Zelen, M. (1989). Rethinking centrality: Methods and examples. *Social Networks*, 11(1):1–37.

Thilakarathna, K., Viana, A. C., Seneviratne, A., e Petander, H. (2013). Mobile social networking through friend-to-friend opportunistic content dissemination. Em *ACM MobiHoc*, pp. 263–266.

Uppoor, S. e Fiore, M. (2011). Large-scale urban vehicular mobility for networking research. Em *IEEE VNC '11*, pp. 62–69.

Wehmuth, K. e Ziviani, A. (2013). DACCER: Distributed assessment of the closeness centrality ranking in complex networks. *Computer Networks*, 57(13):2536–2548.

Yim, J., Ahn, H., e Ko, Y.-B. (2016). The betweenness centrality based geographic routing protocol for unmanned ground systems. Em *IMCOM*, pp. 74:1–74:4.

Zhang, D. e Sterbenz, J. P. G. (2014). Modelling critical node attacks in MANETs. *LNCS Series*, 8221:127–138.

Trilha Principal do SBRC 2017
Sessão Técnica 14
Segurança em Redes I

How to improve monitoring and auditing security properties in cloud storage?

Carlos André Batista de Carvalho[1,2], Nazim Agoulmine[3], Miguel Franklin de Castro[1], Rossana Maria de Castro Andrade[1,*]

[1]Graduate Program in Computer Science (MDCC),
Group of Computer Networks, Software Engineering, and Systems (GREat),
Federal University of Ceará (UFC)

[2]Computer Science Department, Federal University of Piauí (UFPI)

[3]IBISC Laboratory, University of Evry (UEVE)

candrebc@ufpi.edu.br, nazim.agoulmine@ufrst.univ-evry.fr,

{miguel, rossana}@ufc.br

Abstract. *A cloud storage service implements security mechanisms to protect users' data. Moreover, due to the loss of control over the cloud infrastructure, auditing and monitoring mechanisms are used to detect violations of security properties, increasing the trust and transparency in cloud services. However, there are flaws in existing solutions to ensure integrity, freshness and write-serializability properties. Then, we propose a monitoring and auditing mechanism to verify these properties, allowing to detect violations that are not identified by other solutions. Colored Petri Nets (CPNs) are used to model and validate the proposed mechanism. As results, the provider cannot deny the detected violations, and attacks are detected in real-time, except collusion attacks, identified only in our auditing phase.*

1. Introduction

Cloud computing is a distributed computing paradigm that enables the sharing of computational resources between many clients. It is possible to reduce the infrastructure costs by contracting a public cloud provider and paying only for the consumed resources. Besides, the services' scalability allows the dynamic allocation of resources, in accordance with customers' needs. However, this technology comes with the main drawback of losing control over the cloud infrastructure. Therefore, individuals, companies and organizations are resisting to adopt public clouds, due to concerns about security and privacy [Luna et al. 2015].

Cloud providers have developed security mechanisms based on frameworks and security guidelines elaborated by standardization bodies, such as ISO (International Organization for Standardization), NIST (National Institute of Standards and Technology) and CSA (Cloud Security Alliance) [Luna et al. 2015]. However, their customers have a limited view of the security, requesting more transparency with mechanisms that provide service security guarantees [Luna et al. 2015]. In this context, scientific

*Researcher Scholarship - DT Level 2, sponsored by CNPq

research has been realized to develop solutions that improve the trust in cloud providers [Bamiah et al. 2014]. It is important to highlight the audit and monitoring mechanisms to prove the security and allow the violation detection [Popa et al. 2011, Hwang et al. 2014a, Tiwari and Gangadharan 2015]. These mechanisms can be used by a solution of Service Level Agreement (SLA) management to specify and ensure security properties [Carvalho et al. 2017a].

Among security concerns (*e.g.*, data loss and leakage, resource location, service disruption and multi-tenancy issues), [Rong et al. 2013] stress that the big concern is the assurance of the security in cloud storage. In this context, it is important to prove that security properties are achieved. Usually, confidentiality, integrity and availability are the required security properties [Zou et al. 2015], but the literature highlights other properties related to secure cloud storage, such as retrievability [Tiwari and Gangadharan 2015], freshness [Popa et al. 2011], write-serializability [Popa et al. 2011] and location [Albeshri et al. 2014]. The use of cryptography is fundamental to offer security, and to prevent, for example, data leakage. However, it is not possible to avoid all kind of attacks, and the service monitoring is essential to identify an attempted attack and to block it.

Normally, the existing solutions are based on audit mechanisms to detect if security properties were violated. The problem with this approach is that violations are detected after an attack, only when the audit is performed. Although the provider may be penalized, maybe it is not possible to recover the damage. Besides, there is the cost of the audit mechanisms that introduce an overhead in the system, and are performed by a Third-Party Auditor (TPA). Then, it is interesting the development of mechanisms to detect violations in real-time, allowing to avoid an attack or to reduce the damage [Hwang et al. 2014a].

Besides, in some solutions, the security assessment is not formally presented, resulting in security flaws. For example, in our preliminary study [Carvalho et al. 2016], the Cloudproof [Popa et al. 2011] was modeled and evaluated using Colored Petri Nets (CPNs). As a result, it was possible to identify scenarios in which security violations were not detected by Cloudproof. Therefore, the design of mechanisms to guarantee security properties is still challenging.

In this paper, we propose a mechanism to verify integrity, freshness, and write-serializability, allowing to detect violations that were not identified by existing solutions. The provider cannot deceive this mechanism, denying a detected violation. In addition, the proposed mechanism will verify security properties in real-time. However, we identify an attack in which the audit is still necessary to detect it. In order to demonstrate the security and guarantee these properties, we model and validate the proposed mechanism using CPNs.

The rest of this paper is organized as follows. In Sections 2 and 3, we discuss, respectively, the background related to secure cloud storage and the related work. The proposed mechanism is described in Section 4 and evaluated in Section 5. Lastly, Section 6 presents the final considerations and future work suggestions.

2. Secure Cloud Storage

In this section, we show an example of a storage service that combines security mechanisms to offer security, transparency and trust in this service. We also describe the security properties and the threats inherent to our monitoring and auditing mechanism.

2.1. An example

A cloud storage service must, at least, allow users to create, read, update and delete files. The security of the stored data and file sharing are common requirements of this type of service. Then, the secure storage service has to include an access control mechanism, allowing the definition of permissions to each file. Figure 1 presents an overview and the stakeholders of this service in which a cloud customer can purchase this storage service, and define permissions for cloud users to perform cloud transactions. The cloud transactions are performed by a cloud provider and managed by a broker, and the security properties monitored by the users and audited by a Third-Party Auditor (TPA).

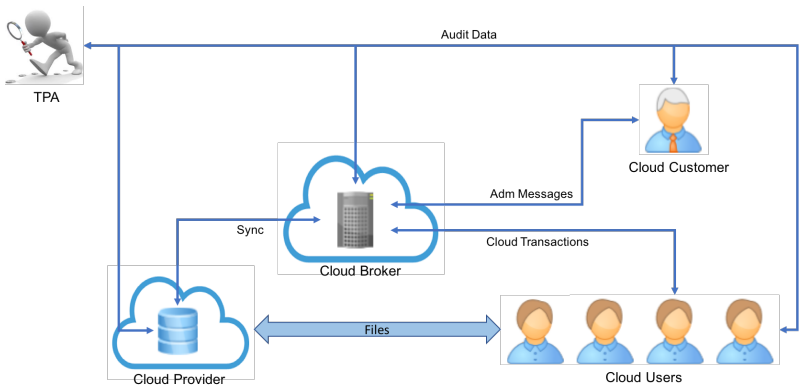


Figure 1. Storage service overview

An access control mechanism protects the service against unauthorized reading and data modification. Besides, a cloud customer can also change files permissions, granting or revoking privileges to users. However, access control and other security solutions are not analyzed in this paper, focusing in verification of security properties. Thus, we assume that the access control mechanism is secure so that only authorized users can perform cloud transactions.

In monitoring, a user analyzes their logs and the messages exchanged during a cloud transaction to verify security properties. When a violation is detected, a TPA can perform an audit to avoid that a user falsely accuses the cloud provider. An audit is also performed to detect violations that cannot be identified in real-time. In order to audit the service, the TPA receives the logs from the provider, broker and users. In addition, the involved entities must verify the messages sent through the service. Thus, it is possible to detect external attacks.

The broker and the cloud provider can be deployed in the same cloud infrastructure. However, when different infrastructures are used, it is possible to improve the security (*e.g.*, deploying the broker in a private cloud), and to store files in multiple providers. The use of multiple providers improves the service availability [Celesti et al. 2016], but this is out of the scope of our work.

2.2. Security Properties

Security mechanisms must be designed to protect a user against existing threats, holding the requested security properties. In this research, we analyze the integrity, freshness and write-serializability of the data stored in the cloud.

Integrity violations occur when data are modified or created by unauthorized entities, resulting in data corruption. Normally, the integrity verification is based on hash functions or messages authentication codes (MACs), and the security of these cryptographic primitives is attested by the scientific community. The freshness indicates the read of the updated file, and the write-serializability controls the writing order. So, the write-serializability ensures that the new version of a file overwrites the last version of it. The verification of freshness and write-serializability is inherent in the operations of, respectively, reading and writing. A writing can be performed to update a file or create a new one. [Popa et al. 2011] suggest writing an empty file to remove it. Due to the possibility of the provider denying a detected violation, it is necessary to sign all logs, providing non-repudiation of the cloud transactions [Hwang et al. 2014b].

The existing solutions analyze the logs to detect some violation. However, the provider can restore the stored files from a backup when some problem occurs, without modifying the logs. In this case, the provider can overwrite the old version of a file, without being detected in write-serializability verification [Carvalho et al. 2016]. Then, before updating a file, it is necessary to check if the current version of the stored file is the same as that indicated by the broker. In order to verify this, it is possible to read the previous file before each writing. However, this approach is not efficient, and the Proof of Retrievability (PoR) schemes can be used to verify the loss of a file, without recover all file [Tiwari and Gangadharan 2015, Albeshri et al. 2012].

Besides data corruption and loss, the data leakage is an important issue of cloud storage [CSA 2013]. Thus, the access control of the files should be only allowed for authorized users, providing confidentiality. Although the confidentiality is not addressed at this moment, a complete solution must include an access control mechanism. On the other hand, it is possible to occur a data leakage because the logs are accessed by the broker, provider and TPA. Then, these logs cannot have sensible information to preserve the users' privacy.

2.3. Threats and Attacks

The monitoring and auditing will increase the transparency of cloud storage services, enabling the detection and prove of violations. Thus, it is necessary to identify the attacks scenarios and evaluate if the proposed mechanism detect them. In this section, we describe the threats and attacks that can affect security properties. The focus is to identify the malicious acts of the provider, broker and external attackers.

In accordance with the Doley-Yao model [Dolev and Yao 1983], an external adversary can impersonate any entity (user, broker or provider), creating or modifying messages. However, a corruption attack is easily detected when the integrity is verified. Although an attacker cannot understand the content of a message, he/she can store it and send this message during a replay attack. Thus, all entities must include verifications to detect corruption or replay attacks. Besides the attackers, the broker and provider can have malicious behaviors, and the users must identify them.

A malicious provider can: i) send an outdated data; ii) write files out of order; iii) confirm a writing transaction, without committing it; and iv) perform a transaction of an unauthorized user. The last malicious behavior is addressed by the access control mechanism because unauthorized users do not have the credentials to read a file or create an uncorrupted file. A replay attack can result in writing files out of order. In a rollback attack, the provider restores the system to a previous state [Hwang et al. 2014b]. Consequently, a user receives an outdated file, or the files are written out of order, and the proposed mechanism detects the violation.

Likewise, a rollback in the broker results in the loss of the attestation of the last transaction. In this case, the provider can inform the stored attestations to prove that the problem occurs in the broker. It is important also to analyze the collusion attacks, where the broker helps the provider to deceive the violation detection mechanism. For example, during a rollback attack, the provider restores the system to a previous state and, the broker informs an old attestation in order to avoid the violation detection.

Besides, in another possible scenario, a broker and provider can restore the system to different states. In this case, if the broker restores to a state older than the provider, the violation in the provider could not be detected, reporting that the problem was only in the broker. Thus, an audit must use the information about last transactions of each user to identify the violation of the provider.

3. Related Work

Our analysis of literature reveals solutions that address security issues in cloud storage, especially related to data corruption, loss and leakage. Normally, the solutions combine security mechanisms to protect the system against several threats, holding security properties. In this section, we focus on analyzing the mechanism to verify security properties of the related work although other mechanisms are used.

For example, CloudProof combines an access control mechanism to provide confidentiality and a mechanism to verify integrity, freshness and write-serializability [Popa et al. 2011]. The cloud transactions are performed by CloudProof, using a protocol to get or put blocks in cloud storage, and attestations are used to record each read or write transaction. During each transaction, the integrity is verified, but violations of freshness and write-serializability are detected only in auditing, after sorting the attestations sent by the users. Due to the signature of the attestations, it is not possible to falsely deny or accuse the occurrence of a violation. The audit is periodically carried out, and some blocks can be randomly selected for auditing, reducing the overhead. Besides, the CloudProof can detect replay attacks.

Due to the informal discussion of the theorems presented by [Popa et al. 2011], we decided to use CPNs to model and validate CloudProof, in order to identify ambiguities and security flaws on it [Carvalho et al. 2016]. As results, we can highlight mainly the lack of management of concurrent requests by users and the identification of scenarios in which security violations were not detected. These scenarios exist because the detection of violations is based only on the chain of attestations, without considering the stored data. For example, a malicious provider can inform a writing attestation, without performing the writing in cloud storage. If the next transaction is a read, the violation can be detected, but this malicious behavior is not identified when a writing transaction is

executed immediately after the first one. Then, it is necessary not only verify the chain of attestations but also if each write was rightly executed.

In another interesting study, [Hwang et al. 2014b] propose a protocol that detects rollback attacks although the attestations are stored by the untrusted provider. Besides, the authors propose a scheme to discard old attestations, and another to allow concurrent access. However, the above violation scenario is not detected by this mechanism nor by the proposal of [Hwang et al. 2014a].

On the other hand, this scenario does not occur in other solutions because the reading is always required before each writing [Albeshri et al. 2012, Tiwari and Gangadharan 2015, Jin et al. 2016]. Nevertheless, this approach is not efficient in a multi-user environment, because a file is blocked until a user finishes its reading, modifying and writing. Although the write-serializability is ensured, it is not informed that the previous reading is mandatory. In addition, the architectures, proposed in these papers, also include mechanisms to prove the retrievability, ensuring even scarcely accessed files were not lost.

The audit is mandatory to verify the retrievability, but freshness and write-serializability should be monitored by real-time mechanisms [Jin et al. 2016, Hwang et al. 2014a]. However, there is an error in the designed protocol by [Jin et al. 2016]. In this protocol, *"only the owner who holds the secret master key can rotate the root signing key to a new version"*, and, at the same time, the users rotate the **root signing key** when files are updated.

Besides, [Hwang et al. 2014a] specify the use of a trustworthy synchronization server to inform the users the current state of the provider. This assumption is acceptable when this server is deployed in a private cloud, and therefore the collusion attack was not analyzed. We studied the effect of the use of an untrusted broker and concluded that an audit is yet necessary (see Section 5). The audit can also be performed to solve any contestation due to the non-repudiation of the cloud transactions [Hwang et al. 2014b].

Table 1. Comparison of related work

Work	Security Properties			Real-time verification	Collusion attack
	Integrity	Freshness	Write-serializabilty		
Popa et al. 2011	Yes	Yes	Partially	Only integrity	Unfeasible
Hwang et al. 2014b	Yes	Yes	Partially	Only integrity	Unfeasible
Hwang et al. 2014a	Yes	Yes	Partially	Yes	Undetected
Albeshiri et al. 2012	Yes	Yes	Yes (inefficient)	Only integrity	Unfeasible
Tiwari and Gangadharan 2015a	Yes	Yes	Yes (inefficient)	Only integrity	Unfeasible
Jin et al. 2016	Yes	Yes	Yes (inefficient)	Fail*	Unfeasible

* The freshness verification depends on the knowledge of the lastest root signing key. However, the authors do not prove how the users obtain this key for real-time verification.

In Table 1, the related work are compared. It is important to highlight that the success of the write-serializability verification depends on the previous reading of a file. Besides, in some studies, the collusion attacks are not possible because there is no broker (or other third entity) participating of cloud transactions. However, the real-time detection of freshness violations is not also possible, making the auditing necessary to detect them.

Lastly, it is interesting to mention about the evaluation of related work. The trend in the literature is the use of standalone PCs to evaluate the cost of proposed algorithms together with an informal security discussion. The single exception is the Cloudproof that was deployed in the Azure infrastructure [Popa et al. 2011]. However, the performed experiments do not indicate the absence of security flaws.

4. Proposal

A secure storage service should include a mechanism to monitor and audit the security properties, enabling the violation detection. We propose a mechanism to verify integrity, freshness and write-serializability of the data stored in the cloud. This mechanism must be combined with an access control mechanism to provide a solution suitable for the file sharing environment. Existing work already combines these solutions, but limitations were found in the violation detection mechanism. Our mechanism fixes the found security flaws and enables the detection in real-time when a trustworthy broker is used. Even so, the audit is necessary to solve any contestation and to identify collusion attacks.

The files are ciphered to provide confidentiality, and an access control mechanism allows the client to define and update the files' permissions. Usually, in related work, the Access Control Lists (ACLs) are used to specify the permissions, and the key management is based on Broadcast Encryption and Key Rotation [Popa et al. 2011, Jin et al. 2016]. Thus, the users can obtain keys for reading and writing from file's metadata if they have permission. Although the access control is verified during the cloud transactions, we assume the correctness of its functioning and do not treat the procedures to grant and revoke permissions.

Thus, we can focus on the verification of security properties that is performed during reading and writing transactions. The broker manages the transactions, being responsible for the control of concurrent transactions and for sending of the last attestation. Using this attestation, a user verifies the integrity and freshness of the read file, or prepare a writing request that complies with the write-serializability.

The elements of an attestation are: **UserID**, **UserLSN**, **FileID**, **FileVersion**, **FileHash**, **TransactionType**, **ChainHash** and **Signatures**. The UserLSN represents the last sequence number used by each user, enabling to detect replay and rollback attacks. The FileVersion is essential to verify the freshness and write-serializability, and the FileHash to check the integrity. The type of the transaction (i.e., reading or writing) indicate whether the expect FileVersion must be replaced during an auditing. The ChainHash is used to build the chain of attestations and is computed over the data of the current attestation and the ChainHash of the previous one. Lastly, all involved entities (i.e., broker, user and provider) sign the attestation for non-repudiation purposes.

It is essential to specify the management of concurrent transaction. The broker must allow simultaneous readings of the same file, blocking only during a writing. This blockade is necessary to avoid the reading of the previous version of a file and the requests of writings with the same FileVersion. On the other hand, the concurrent access to different files is allowed because there is no possibility of violation resulting from such access.

The broker must store the last attestation of each file and send it to users to verify the security properties. Besides, the users store the attestations of their last transactions

with each file and report a violation if a file version informed by the broker is older than the one stored by the user. It is mandatory the sending of all attestations of each user to identify other scenarios of collusion attacks because not all attacks are detected in real-time. Therefore, the proposed mechanism defines the building of one chain of attestation per file.

On the other hand, it is not necessary that each user manages one UserLSN per file so that a single UserLSN per user is enough to verify security properties. The broker and provider also store the current UserLSN of all users in order to detect replay attacks. With this approach, it is not possible to probabilistically choose the files to be audited as suggest by [Popa et al. 2011] because, if some file is ignored, some UserLSNs will not be found. Anyway, all files must be audited to detect all violations scenarios.

Before the audit, the provider must send all attestations to the TPA, and the broker and users send their last attestations. For each file, the TPA builds the chain of attestations, ordering them, and analyzes the sequence of the file's versions to prove that no violation occurs. Besides, it is also check the presence of all UserLSNs considering all chains of attestation. The attestations sent by users and broker are used to verify whether the provider hid some transaction. Otherwise, the TPA report a violation. Due to the signature of attestations, no entity can deny a violation. After the audit, the attestations can be discarded, except the last attestation of each file that will be chained with the attestations of the new epoch[1].

Thus, this mechanism includes auditing procedures to verify the assurance of security properties during cloud transactions. In addition, the monitoring of the properties is performed during the clouds transactions in order to detect violations and attacks in real-time. Due to space constraint, only the protocol to perform read transactions is detailed in Figure 2. Besides the attestations, the exchanged messages are signed, and a receiver verifies the authenticity of the sender. Thus, any entity can detect attacks when invalid signatures are used.

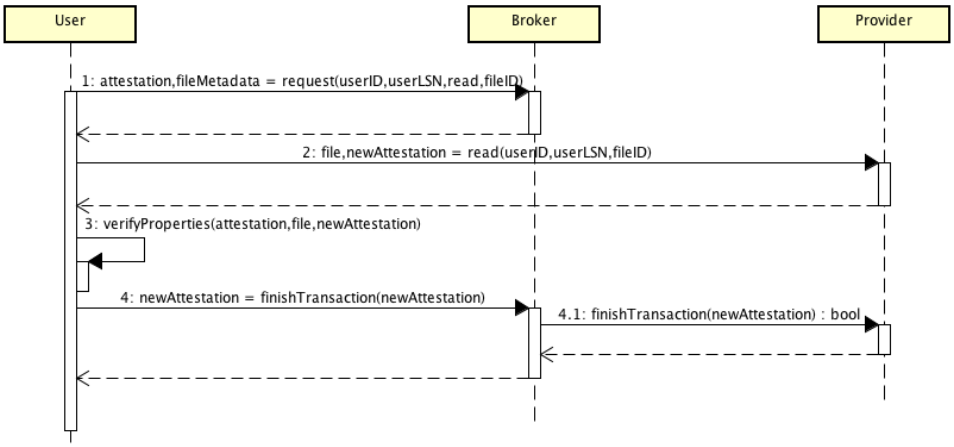


Figure 2. Reading a file

Before a user requests the reading of a file, a user sends a message to broker in order to receive the last attestation and the file's metadata. Next, this user deciphers the

[1]The period between two consecutive audits is called **epoch** [Popa et al. 2011]

encryption key and request the file to the provider. The provider sends the file and the new attestation signed by it. Then, the user verifies the integrity and freshness, using FileHash and FileVersion respectively. If no violation is found, the user and the broker sign the new attestation and send it to the provider. The broker and the user overwrite their last attestation.

During a writing, a user extracts the signature key to sign the new file. This signature is placed in the FileHash field of the attestation. The verification key is public, and the provider checks the signature, testing if the user is authorized before committing a writing. The provider also checks the FileVersion for write-serializability purposes.

One flaw of existing work is the verification of write-serializability based only on attestations. Thus, there are undetected violation scenarios, making it necessary the verification of the file version really stored by the provider. In order to check it, we use a PoR scheme, but its evaluation is outside the scope of this paper because we do not propose a PoR scheme. Besides, it was concluded, in related work, that the other option is more costly due to the full reading of the files [Albeshri et al. 2012, Tiwari and Gangadharan 2015]. In writing protocol, the PoR is performed before a writing so that a user makes a challenge and verifies the proof sent by the provider. The keys and tags used during this step are also stored in file's metadata. This metadata must be updated, with the tags related to the new file. This approach enables to properly verify the write-serializability.

It is worth highlighting that the malicious actions of the broker, and collusion attacks, aren't addressed in related work. During a collusion attack, the broker sends the attestation and metadata in accordance with an old file stored by the provider. Thus, it is possible, for example, to receive and accept an outdated file, because the attestation wrongly indicates that the file is up-to-date. On the other hand, sometimes, a user can detect the violation based on his/her last attestation. For example, a user requested a writing and stored the attestation of this transaction. Next, the user can detect a violation if he/she reads an old version of the file. However, this user can have his/her permissions revoked and does not perform new transactions. Thus, if the broker is compromised, the users may not detect, in monitoring, the violations because they do not have the correct information about the current state of the system, making the auditing indispensable to identify the undetected violations through historical analysis of cloud transactions.

Any involved entity can detect an external attack (e.g., replay attack) and cancel the unauthorized transaction, but a malicious behavior of the provider and/or broker results in a security violation. In this case, a penalty and a recovery procedure can be applied as specified in an SLA. For example, if an integrity violation is detected, it is possible to restore the most up-to-date version in a backup, reducing the damage. Therefore, the proposed mechanism monitor and audit security properties, detecting violations of integrity, freshness, and write-serializability.

5. Modeling and Validation

The use of formal methods is a good approach to model and validate the proposed mechanism, given the difficulty of demonstrating the security only with experiments in cloud infrastructures. We use the CPN Tools[2] to model this mechanism, because of

[2]http://cpntools.org

the ease of use of this tool and its extensive documentation, as well as the suitability of evaluating security protocols [Carvalho et al. 2016]. In this section, we present the modeling of the proposed, as well as, the scenarios to evaluate the robustness of this mechanism against existing threats and flaws found in related work. The modeled CPN includes the protocols for auditing and for reading and writing files in order to analyze the detection of violations.

A simplified CPN of this mechanism is available at https://sites.google.com/site/candrebc/scss.zip[3] and the overview of this modeling is shown in Figure 3. The simplifications do not change it functioning and include: i) the storage of a single file; ii) the execution of a single transaction each time; iii) the representation of the chain hash as a sequence number; and iv) the assumption that all messages are authentic. The modeled CPN is parametrized so that several users can be easily added making this mechanism theoretically scalable. Besides, the modeling indicates a violation when the system reaches one violate state, represented by the presence of tokens in error places of the CPN. These simplifications facilitate the analysis of the modeling due to the reduction of its complexity and of the space state.

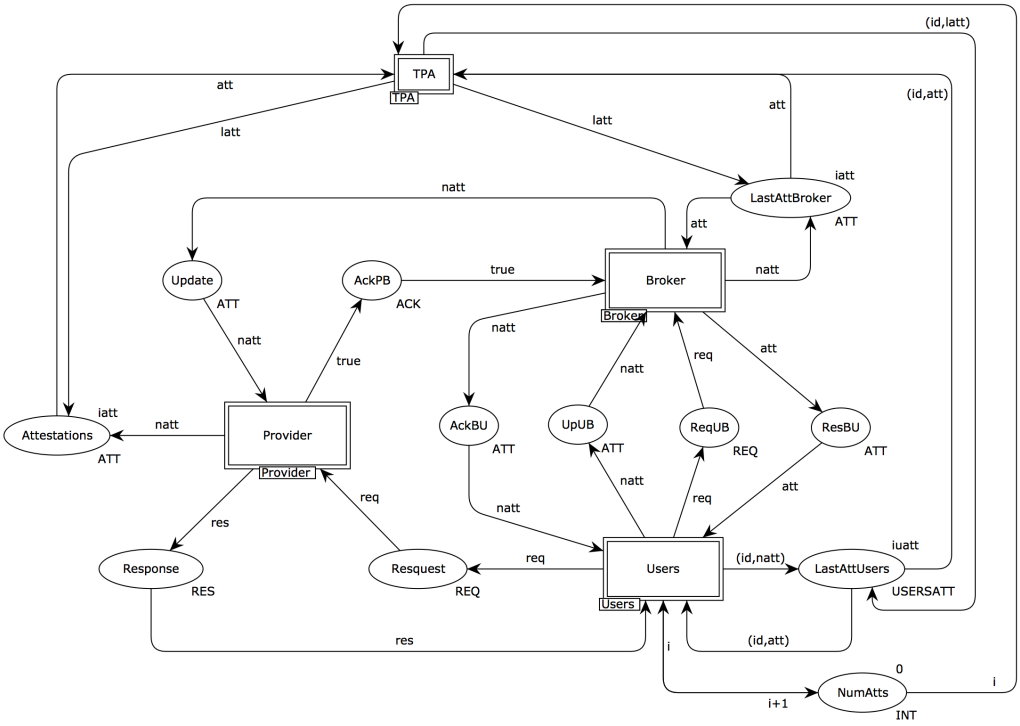


Figure 3. Overview of the simplified CPN

This modeling represents the correct behavior of the involved entities and simulations were realized to verify the correctness and to evaluate the modeled mechanism. Thus, none violation is detected because the analysis of the space state reveals that no token is stored in error places in any possible state. This result is expected when the entities are trustworthy, and this mechanism demonstrates the assurance of security properties of the modeled storage service.

[3]The original file can be used for proper analysis and simulations. A *.pdf* file is also available.

In order to validate the proposed mechanism, it is necessary to demonstrate that all possible attack is detected, especially that resulting from malicious actions by the provider and broker. These actions result in violations that must be detected by the users or the TPA. Then, we modeled malicious behaviors of the provider and broker in different CPNs, simulating each violation scenario. The specification of two users in each modeled CPN was enough to detect the violations. We also define a special place to indicate that the provider or broker execute malicious actions. Thus, every time that a violation is triggered, the mechanism should detect it.

Table 2 presents the results of the validation, reporting that the violations are always detected, in accordance with the attacks enumerated in Subsection 2.3. Although the users can identify previously the collusion attacks based on their last attestation that indicates the attestation sent by the broker is old, we inform in this table the worst case of violation detection.

Table 2. Violation detection

Provider	Broker	Detection time
Honest	Send an outdated file	Monitoring
Honest	Write out of order	Monitoring
Honest	Do not commit a writing	Monitoring
Send an old attestation	Send an outdated file	Auditing
Send an old attestation	Write out of order	Auditing
Send an old attestation	Do not commit a writing	Auditing
Send an old attestation	Honest	Monitoring

We modeled the rollback attack that restores the provider to a previous state of the system. It is possible to rollback only the files or the files and the attestations. In both cases, the sending of an outdated file is detected by the users when the broker is honest and informs the expected version of the file. However, if the broker colludes with the provider, sending an old attestation, the TPA will identify the violation.

If it is requested a writing after the rollback, the provider can only write a file out of order because the change of the file version results in an integrity violation. The absence of some file versions is detected by the TPA when ordering the attestations only if the provider restores to previous files and attestations. However, if the broker is honest, the write-serializability violation is detected in real-time when the verification of retrievability is included in all writing transactions. This verification fixes security flaws found in related work, being an important improvement of the violation detection mechanism. For simplicity, this step is represented in our modeling by the reading of the file, but it must be replaced by a PoR scheme as defined in our proposal.

In order to verify the retrievability, we modeled a provider that confirms the writing of the second version of a file, without committing it (i.e., the provider send the writing attestation, but keep only the first version of the file). It is visible that the violation is detected if the next transaction is a reading. Otherwise, if the next transaction is a writing, the first version will be overwritten by the third version, violating the write-serializability. Thus, no violation is identified even in the auditing because the attestations are rightly ordered in it. Then, it is necessary to verify what is the version of the file really stored, before requesting a writing. It is interesting to highlight that we

analyzed the possibility of verifying the writing after performing it, but the data loss of a rollback attack could not be detected.

On the other hand, in a collusion attack, the broker informs to a user the same version of the file sent by the provider. In the worst case, the TPA reports the violation because there is a reduction of the file versions, or some UserLSN is absent. The detection of collusion attacks is not addressed by related work and is another improvement of our mechanism.

Loss of data, resulting from the malicious provider that do not commit a writing request, can trigger a freshness or write-serializability violation, depending on whether the next transaction is a reading or writing. In both cases, the violation is detected in a similar manner to the previous scenarios.

Besides, this mechanism detects malicious behaviors of the broker that can inform an old attestation. The users identify this behavior when the honest provider send a newer version of the requested file. The last attestation maintained by the provider is signed by also by the broker and can be used to prove the failure in the broker. The audit is only necessary to demonstrate if the provider also lost some information. If the broker loses the tags used by the PoR scheme, the file must be read to check its version.

External attackers can meddle in the communication to perform illegal transactions, according to Doley-Yao model [Dolev and Yao 1983]. The use of cryptography to sign the messages and the verification of the UserLSN make these attacks unfeasible. In addition, the access control mechanism is secure so that an unauthorized user cannot obtain the encryption and signatures keys. Thus, an unauthorized reading is not possible, and the provider can identify an unauthorized writing, based on the signature of the file. On the other hand, a malicious provider can commit this writing or own a corrupted file. Due to the absence of failures in integrity verification of the related work, we do not specify the execution of illegal transactions in Table 2. However, the resulting violations are detected during the monitoring since the security of cryptographic primitives is attested by the scientific community. In these cases, the integrity violation is detected when a user reads the corrupted file, or the challenge-response fails, during a writing.

We also observed how the modifications in the proposed mechanism affect the violation detection. For example, the users and the broker must keep their last attestations of all files even if a single chain of attestation is used for all files. The real-time detection is improved because the attestation about one file is not overwritten after a transaction with another file. If the UserLSNs is not verified in auditing, violations resulting from the loss of attestations in collusion attacks are not detected.

When analyzing the concurrent access, no violation occurs if simultaneous readings are requested. However, simultaneous writings can result in the commitments with the same FileVersion, or in the absence of some FileVersion if some transaction is not finalized. In addition, if a reading is requested before a writing ends, it is possible to receive the written version or the previous one.

Although the main goal is the detection of malicious acts of the provider, broker and external attackers, our modeling proves that some malicious actions of users are also identified. For example, when requesting a writing operation, a user can inform the wrong

version of a file, and the broker and provider detect this act by knowing the expected version. However, it is indispensable the users rightly inform their last attestations to avoid that some violation is not identified by the auditing.

Lastly, it is important to highlight that the management of transaction by the broker enables the detection of violations in real-time. Our analysis demonstrates that violations are detected in real-time if the broker is honest. The audit is then necessary to solve any contestation and identify collusion attacks.

6. Conclusion

A secure cloud storage service must include a mechanism to monitor and audit security properties. With this mechanism, a provider proves the assurance of these properties, improving the transparency and trust of security storage services. During this research, we identify that existing mechanisms do not detect all security violations and fail in demonstrating the security. Besides, related work does not properly address the real-time detection and collusion attacks.

In this paper, we described a mechanism to verify integrity, freshness and write-serializability, fixing existing flaws. The modeling with CPNs enables a formal validation of the proposed mechanism and proves the need of auditing to detect collusion attacks. In the validation, the attacks were modeled and detected by this mechanism. As proposed in [Carvalho et al. 2017b], a secure cloud storage service must combine this mechanism with an access control mechanism to provide also the confidentiality, protecting the customers against data loss and data leakage while enabling the data sharing. The proposed mechanism can also be used with an SLA solution to provide these properties. However, it is necessary to define the penalties and contingency plan to reduce the damage when a violation is detected.

This paper focuses on improving the violation detection and on security evaluation. As future work, this mechanism could be deployed in a cloud infrastructure to evaluate its functioning in a real scenario and analyzing performance aspects. It is possible to analyze other access control mechanisms (*e.g.*, proxy re-encryption or Attribute-Based Encryption [Thilakanathan et al. 2014]), identifying what is most suitable for secure storage environment. A robust solution should also include mechanisms to address other security properties such as availability and location.

Acknowledgments

This work was partially supported by the STIC-AmSud project SLA4Cloud. Carlos André Batista de Carvalho was also supported by CAPES/FAPEPI Doctoral Scholarship.

References

Albeshri, A., Boyd, C., and Nieto, J. G. (2012). A security architecture for cloud storage combining proofs of retrievability and fairness. In *3rd International Conference on Cloud Computing, GRIDS and Virtualization*, pages 30–35.

Albeshri, A., Boyd, C., and Nieto, J. G. (2014). Enhanced geoproof: improved geographic assurance for data in the cloud. *International Journal of Information Security*, 13(2):191–198.

Bamiah, M. A., Brohi, S. N., Chuprat, S., and lail Ab Manan, J. (2014). Trusted cloud computing framework for healthcare sector. *Journal of Computer Science*, 10(2):240–240.

Carvalho, C. A. B., Andrade, R. M. C., Castro, M. F., and Agoulmine, N. (2016). Modelagem e detecção de falhas em soluções para armazenamento seguro em nuvens usando redes de petri coloridas: Um estudo de caso. In *XIV Workshop de Computação em Clouds e Aplicações (WCGA/SBRC)*, pages 17–30. in portuguese.

Carvalho, C. A. B., Andrade, R. M. C., Castro, M. F., Coutinho, E. F., and Agoulmine, N. (2017a). State of the art and challenges of security SLA for cloud computing. *Computers and Electrical Engineering*. In Press.

Carvalho, C. A. B., Castro, M. F., and Andrade, R. M. C. (2017b). Secure cloud storage service for detection of security violations. In *CCGrid'17 Doctoral Symposium*. Accepted for publication.

Celesti, A., Fazio, M., Villari, M., and Puliafito, A. (2016). Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems. *Journal of Network and Computer Applications*, 59:208–218.

CSA (2013). The notorious nine: Cloud computing top threats in 2013. Technical report, Top Threats Working Group.

Dolev, D. and Yao, A. C. (1983). On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208.

Hwang, G.-H., Huang, W.-S., and Peng, J.-Z. (2014a). Real-time proof of violation for cloud storage. In *CloudCom'14*, pages 394–399.

Hwang, G.-H., Huang, W.-S., Peng, J.-Z., and Lin, Y.-W. (2014b). Fulfilling mutual nonrepudiation for cloud storage. *Concurrency and Computation: Practice and Experience*.

Jin, H., Zhou, K., Jiang, H., Lei, D., Wei, R., and Li, C. (2016). Full integrity and freshness for cloud data. *Future Generation Computer Systems*.

Luna, J., Suri, N., Iorga, M., and Karmel, A. (2015). Leveraging the potential of cloud security service-level agreements through standards. *IEEE Cloud Computing Magazine*, 2(3):32 – 40.

Popa, R. A., Lorch, J. R., Molnar, D., Wang, H. J., and Zhuang, L. (2011). Enabling security in cloud storage slas with cloudproof. In *USENIXATC'11*.

Rong, C., Nguyen, S. T., and Jaatun, M. G. (2013). Beyond lightning: a survey on security challenges in cloud computing. *Computers and Electrical Engineering*, 39(1):47–54.

Thilakanathan, D., Chen, S., Nepal, S., and Calvo, R. A. (2014). Secure data sharing in the cloud. In *Security, Privacy and Trust in Cloud Systems*, pages 45–72. Springer.

Tiwari, D. and Gangadharan, G. (2015). A novel secure cloud storage architecture combining proof of retrievability and revocation. In *ICACCI'15*, pages 438–445.

Zou, H., Qian, Y., Zhao, Y., and Ding, K. (2015). The design and implementation of data security management and control platform. In *ATIS'15*, pages 368–378.

Um Algoritmo Não Supervisionado e Rápido para Seleção de Características em Classificação de Tráfego *

Martin Andreoni Lopez[1,2], Antonio G. P. Lobato[1],
Diogo Menezes F. Mattos[1,2], Igor D. Alvarenga[1],
Otto Carlos M. B. Duarte[1], Guy Pujolle[2]

[1]GTA / PEE-COPPE / UFRJ – Brasil
[2]LIP6/CNRS (UPMC Sorbonne Universités) – França

{martin, antonio,diogo,alvarenga,otto}@gta.ufrj.br
{guy.pujolle}@lip6.fr

Abstract. *Security applications such as anomaly detection and attack mitigation need real-time monitoring to reduce risk. Information processing time should be as small as possible to enable an effective defense against attacks. In this paper, we present a fast and efficient feature-selection algorithm for network traffic classification based on the correlation between features. For the evaluation of the algorithm, we run the algorithm against a dataset containing more than 16 types of threats, in addition to normal traffic. The presented algorithm chooses an optimized subset of features that improves accuracy by more than 11% with a 100-fold reduction in processing time when compared to traditional feature selection and reduction algorithms.*

Resumo. *Aplicações de segurança como a detecção de anomalias e a mitigação de ataques precisam de monitoramento em tempo real para a diminuição dos riscos. Os tempos para o processamento das informações devem ser os menores possíveis para habilitar elementos de defesa. Este artigo apresenta um algoritmo rápido e eficiente de seleção de características para a classificação de tráfego baseado na correlação entre características. Para a avaliação do algoritmo, é utilizado um conjunto de dados contendo mais de 16 tipos de ameaças, além de tráfego normal. O algoritmo desenvolvido escolhe um subconjunto otimizado de características que melhora a acurácia em mais do 11% com redução de até 100 vezes do tempo de processamento, quando comparado com algoritmos tradicionais de seleção e redução de características.*

1. Introdução

É fundamental entender o tipo, o volume e as características intrínsecas de cada fluxo, a fim de manter a estabilidade, a confiabilidade e a segurança de uma rede. Monitores de redes eficientes permitem ao administrador um melhor entendimento do comportamento da rede [Hu et al. 2015]. O monitoramento da rede possui diversos níveis de complexidade. Desde uma simples coleção de estatísticas de utilização de link até complexas análises de camadas superiores para a realização de sistemas de detecção de intrusão (*Intrusion Detetection System* – IDS), otimização de desempenho ou depuração de protocolos. Os sistemas atuais de monitoramento de rede como `tcpdump`, NetFlow, SNMP, Bro [Paxson 1999], Snort [Roesch 1999], entre outros, são executados em um único servidor e não são capazes de atingir as velocidades de tráfego atuais de forma escalável [Vallentin et al. 2007]. O IDS Bro, por exemplo, evoluiu

*Este trabalho foi realizado com recursos do CNPq, CAPES, FAPERJ e FINEP.

para a utilização em agrupamentos (*clusters*), a fim de atingir 10 Gb/s. O IDS Suricata foi proposto como uma melhoria sobre o IDS Snort, pois utiliza GPU (*Graphical Processing Unit*) para paralelizar a inspeção de pacotes.

No monitoramento de tráfego, dados chegam em tempo real, originados de fontes heterogêneas de informação [Bar et al. 2014], como a captura de pacotes de redes ou os *logs* de sistemas e aplicações. Esse tipo de monitoramento em tempo real por fluxo é caracterizado por uma sequência de eventos infinita, abstraídos em uma estrutura de dados chamada de tuplas, que chegam continuamente [Stonebraker et al. 2005]. Um dos problemas deste tipo de monitoramento é o volume de dados gerados. Mesmo redes de velocidades moderadas podem gerar uma grande quantidade de dados. Monitorar um link *ethernet gigabit*, por exemplo, funcionando a uma taxa de 50% de utilização, gera um *terabyte* de dados em questão de horas.

Uma forma de otimizar os métodos de processamento é através da utilização de algoritmos de aprendizado de máquina. As técnicas de aprendizado de máquina são adequadas para o processamento de grandes massas de dados (*big data*), já que, com maiores amostras para o treinamento, as técnicas tendem a melhorar a sua efetividade [Mayhew et al. 2015]. No entanto, com grandes volumes de dados, esses métodos apresentam maiores atrasos devido ao alto consumo de recursos computacionais. Esse atraso é uma desvantagem para esses métodos, já que com as grandes massas de dados, o processamento deve ser tão rápido quanto possível, a fim de se ter respostas em tempo real. A seleção de características é uma forma de resolver esse problema, já que reduz o número de características analisadas pelos algoritmos de aprendizado de máquina a um subconjunto menor do que o original [Mladenić 2006].

Este artigo apresenta um algoritmo eficiente e rápido de seleção de características para a classificação de tráfego. O algoritmo realiza a seleção de características de modo não supervisionado, ou seja, sem conhecer a classe de cada fluxo *a priori*. Além disso, o algoritmo determina as principais características para fazer uma classificação acurada do tráfego entre as classes normal, negação de serviço e ameaças de varredura de portas. Para a avaliação do algoritmo é utilizado um conjunto de dados[1] criado a partir do monitoramento de tráfego real [Pastana Lobato et al. 2016]. O algoritmo apresentado é comparado com métodos tradicionais, tais como ReliefF [Robnik-Šikonja e Kononenko 2003], Seleção de Características Sequencial (*Sequential Feature Selection* - SFS), a Análise de Componentes Principais (*Principal Component Analysis* - PCA) [Schölkopf et al. 1999] e a Seleção Recursiva de Eliminação por Máquinas de Vetores de Suporte (*Support Vector Machine Recursive Feature Elimination* SVM-RFE) [Guyon et al. 2002]. O algoritmo apresenta melhor acurácia nos métodos de aprendizado de máquinas utilizados, assim como uma melhora no tempo de processamento total.

O restante do artigo está organizado da seguinte forma. A Seção 2 discute os trabalhos relacionados. O conjunto de dados de segurança é apresentado na Seção3. O algoritmo de Seleção de Características é introduzido na Seção 4. Na Seção 5, o algoritmo apresentado é comparado com métodos tradicionais de seleção de características. Por fim, a Seção 6 conclui o trabalho.

2. Trabalhos Relacionados

Existem dois modos de reduzir o número de variáveis a serem analisadas e melhorar o tempo de processamento dos algoritmos de aprendizado de máquinas: redução e projeção de características. O primeiro deles é a seleção de características, no qual um subconjunto das características é selecionado de acordo com a quantidade de informação que cada carac-

[1] O conjunto de dados é disponibilizado através do contato com os autores por *e-mail*.

terística possui. Existem diversas métricas para se medir a quantidade de informação de cada característica, como por exemplo, o ganho de informação ou a correlação entre elas, como no algoritmo apresentado. O segundo modo é a redução de características através da projeção dos dados em um novo espaço vetorial, no qual uma transformação é realizada sobre o conjunto de características original, o que leva o conjunto original para um espaço de dimensão menor, em que as novas características são combinações das originais. No novo espaço, o número de novas características a serem tratadas pelos algoritmos de aprendizado é reduzido através do descarte de componentes menos significativas. Cada característica nesse novo espaço é uma combinação das características originais [Van Der Maaten et al. 2009].

A Análise de Componentes Principais (*Principal Component Analysis* - PCA) é um dos métodos mais populares para a redução não supervisionada de características. A ideia do PCA é projetar as características originais de maneira que as novas características sejam ortogonais, ou seja, independentes. A partir dessa transformação, as características que representam as maiores variâncias dos dados originais são selecionadas. A direção que representa a maior variância projetada é chamada da Componente Principal. Assim, as características geradas são uma combinação linear das componentes principais. Como os componentes são ordenados seguindo uma ordem decrescente de variância, a dimensionalidade do dado original pode ser reduzida, eliminando as componentes com a menor variância. A maior desvantagem das técnicas de projeção de características é a perda de expressividade, já que as características geradas são combinações de outras características e perdem seu significado original.

Uma solução para isso é a utilização de métodos de seleção de características em que um subconjunto das características originais é escolhido. A seleção de características pode ser dividida em três grupos [Mladenić 2006]: *wrapper*, filtragem e embarcados. Nos métodos *wrapper* são usados diferentes classificadores como as Máquinas de Vetores de Suporte (*Support Vector Machine*- SVM), redes neurais, entre outros, para dar uma pontuação a um subconjunto de características. A partir da pontuação, escolhem-se as características que obtiveram a maior pontuação ao serem usadas para classificar os dados. Os métodos de filtragem analisam as propriedades intrínsecas dos dados, também chamadas de metadados, e realizam algum tipo de função para determinar a sua importância. Os métodos embarcados realizam a seleção mediante o uso de classificadores no processo de aprendizado.

Os métodos de filtragem são computacionalmente mais leves e evitam a superespecialização ou *over-fitting*. Esses métodos utilizam alguma heurística para avaliar a relevância da característica dentro do conjunto de dados [Chandrashekar e Sahin 2014]. Um dos métodos mais populares de filtragem é o Relief. Nele, a pontuação é calculada como a diferença de distância da amostra mais próxima de uma mesma classe e a amostra mais próxima de uma classe diferente. Um aspecto negativo desse método é ter de saber as classes do conjunto de dados. O Relief é limitado a duas classes, mas o ReliefF [Robnik-Šikonja e Kononenko 2003] permite a análise de múltiplas classes utilizando a técnica dos k-vizinhos mais próximos.

Os métodos de *Wrapper* utilizam o erro dos classificadores como critério de avaliação da característica. A seleção é considerada como um problema de busca, criando um problema de tipo NP-complexo. Os métodos de *wrapper* apresentam melhores resultados que os métodos de filtragem, porém com alto custo computacional [Ang et al. 2016]. Uma das implementações mais populares do *Wrapper* é o (*Sequential Forward Selection* - SFS). O algoritmo começa com um conjunto vazio S e o conjunto completo de todas as variáveis X. O SFS faz uma pesquisa e gradualmente adiciona variáveis selecionadas por uma função de avaliação que minimiza o erro quadrático médio. Em cada iteração, a variável a ser incluída em S é selecionada entre as restantes em X. A principal desvantagem do SFS é que, ao adicionar uma característica ao

conjunto S, o método não é capaz removê-la se ela tiver menor erro, após a adição de outras.

Os métodos embarcados (*embedded methods*) apresentam um comportamento semelhante aos métodos *Wrapper*, usando um classificador para avaliar a relevância da variável. No entanto, os métodos embarcados realizam a seleção da variável no processo de aprendizado, reduzindo o tempo computacional dos *Wrappers*. Na Seleção Recursiva de Eliminação usando Máquinas de Vetores de Suporte (*Support Vector Machine Recursive Feature Elimination* SVM-RFE) [Guyon et al. 2002], o algoritmo obtém uma pontuação das variáveis baseado no treinamento do SVM com *kernel* linear. A característica com a menor pontuação é removida de acordo com um critério w em modo sequencial de eliminação para trás. O critério w é o valor da decisão do hiperplano no SVM.

O algoritmo apresentado neste artigo é inspirado na Seleção de Características baseada em correlação [Hall 1999] (*Correlation based Feature Selection* - CFS). Hall utiliza a ideia de pontuar as variáveis através da correlação entre as variáveis e a classe de saída. O algoritmo CFS calcula a correlação entre a variável e a classe para obter a importância de cada característica. Assim, o CFS é dependente da informação da classe de saída *a priori*, logo é um algoritmo supervisionado. A dependência da informação da classe de saída a que pertence cada amostra é uma limitação em aplicações como a classificação de trafego de redes, já que não existe um conhecimento *a priori* entre o fluxo e a classe correspondente.

O algoritmo apresentado realiza a seleção de características de modo não supervisionado. O cálculo da correlação entre as variáveis é utilizado para medir a quantidade de informação que cada variável representa em relação às demais variáveis. Dessa forma, o algoritmo apresentado executa com menor tempo de processamento e não necessita conhecer *a priori* a classe de saída. A diferença do CFS, que considera a correlação individual entre variáveis, o algoritmo proposto é independente da classe e considera a soma das correlações.

3. Criação de um Conjunto de Dados de Segurança

Existem poucos conjuntos de dados disponíveis para a avaliação de pesquisas de mecanismos de defesa contra ataques. A pouca disponibilidade desses conjuntos de dados é devido à preocupação com a privacidade e o receio de vazamento de informações confidenciais contidas no corpo dos pacotes [Heidemann e Papdopoulos 2009]. Um dos principais conjuntos de dados disponíveis, o DARPA [Lippmann et al. 2000], é composto por tráfego TCP/IP e dados coletados de sistemas operacionais de uma rede simulada. Já o KDD 99 [Lee et al. 1999] é gerado a partir dos arquivos `tcpdump` do DARPA processados e abstraídos em características. Esses conjuntos de dados contêm ataques simulados e marcados como ataques. A principal crítica, tanto para os dados do DARPA como do KDD, é que o tráfego não corresponde a um cenário de uma rede de computadores real[Tavallaee et al. 2009], já que são resultado de simulações. Além disso, existem dados redundantes que afetam o cálculo dos métodos de classificação dos dados. Ademais, esses conjuntos de dados estão defasados, pois foram simulados há mais de 15 anos [Sommer e Paxson 2010], e muitas aplicações, assim como ataques, sugiram e mudaram durante esse período.

O conjunto de dados utilizado foi criado pelos autores previamente [Pastana Lobato et al. 2016]. Esse conjunto de dados é obtido através da captura de pacotes de tráfego real do laboratório do Grupo de Teleinformática e Automação (GTA) da UFRJ. O tráfego capturado contém comportamento normal e ameaças reais de redes. Depois da captura dos pacotes, os dados são agrupados em uma janela de tempo. O tráfego é armazenado em forma de fluxo, uma sequência de pacotes com o mesmo IP de origem e IP de destino.

Cada fluxo possui 24 características obtidas dos cabeçalhos TCP/IP. A lista completa

Tabela 1: As 24 características obtidas do cabeçalho TCP/IP para cada fluxo.

Número	Característica	Descrição
1	qtd_pkt_tcp	Quantidade de Pacotes TCP
2	qtd_src_port	Quantidade de Pacotes Portas Origem
3	qtd_dst_port	Quantidade de Pacotes Portas Destino
4	qtd_fin_flag	Quantidade de Flags FIN
5	qtd_syn_flag	Quantidade de Flags SYN
6	qtd_psh_flag	Quantidade de Flags PSH
7	qtd_ack_flag	Quantidade de Flags ACK
8	qtd_urg_flag	Quantidade de Flags URG
9	qtd_pkt_udp	Quantidade de Pacotes UDP
10	qtd_pkt_icmp	Quantidade de Pacotes ICMP
11	qtd_pkt_ip	Quantidade de Pacotes IP
12	qtd_tos	Quantidade de Tipos de Serviço IP
13	ttl_m	TTL Médio
14	header_len_m	Tamanho Médio dos cabeçalhos
15	packet_len_m	Tamanho Médio dos Pacotes
16	qtd_do_not_frag	Quantidade de Flags “Do Not Frag”
17	qtd_more_frag	Quantidade de Flags “More Frag”
18	fragment_offset_m	Offset Médio do Fragmento
19	qtd_rst_flag	Quantidade de Flags RST
20	qtd_ece_flag	Quantidade de Flags ECE
21	qtd_cwr_flag	Quantidade de Flags CWR
22	offset_m	Offset Médio
23	qtd_t_icmp	Quantidade de Tipos ICMP
24	qtd_cdg_icmp	Quantidade de Códigos ICMP

das características é mostrada na Tabela 1. Ataques de Negação de Serviço (*Denial of Service* - DoS) e ameaças de varredura de porta (*Probe*) compõem as duas classes de comportamento anormal. O conjunto possui sete tipos de DoS e nove tipos de ameaças de varredura.

3.1. Descrição das Ameaças no Conjunto de Dados

No **ataque LAND** (*Local Area Network DoS*) o atacante manda um pacote TCP com etiqueta SYN com o endereço IP falsificado, contendo o endereço e porta da vítima como o endereço e porta de origem e destino. Assim, o sistema responde com um pacote SYN-ACK para ele mesmo, criando uma conexão vazia que permanece aberta até o temporizador estourar. Logo, o sistema entra em um laço enviando respostas para ele mesmo até eventualmente colapsar. ***Nestea*** é um ataque que afeta aos servidores Linux e o ataque ***teardrop*** é o semelhante para servidores Windows. O ataque consiste em explorar uma vulnerabilidade no protocolo TCP/IP ao enviar pacotes fragmentados para a vítima. Quando a soma do deslocamento e tamanho de um pacote fragmentado diferem daquele do próximo pacote fragmentado, os pacotes se sobrepõem e a vítima tenta montar o pacote entrando em uma negação de serviço.

Os ataques por inundação (*flooding*) enviam uma quantidade anormal de pacotes sem esperar respostas. Logo, a vítima, ao tentar processar todos os pacotes, não consegue atender a todas as requisições a tempo e provoca a negação de serviço. No conjunto de dados usado, existem a **inundação por SYN**, a **inundação por ICMP**, e a **inundação por UDP**. Na inundação por SYN são enviados múltiplos pacotes com endereços de origem falsificados para manter co-

nexões abertas e assim estourar os recursos da vítima. Na inundação por ICMP, também são enviados múltiplos pacotes ICMP sem esperar a resposta da vítima. A inundação por UDP tem uma filosofia diferente das anteriores, pois o UDP não é orientado à conexão. Nesse ataque, são enviados pacotes com o mesmo endereço, mas com diferentes portas, com o objetivo de saturar a largura de banda.

O ataque de ***Smurf***, de forma semelhante a inundação por ICMP, utiliza um pacote *Echo Request* (ICMP) com o endereço de difusão (*broadcast*) da rede como endereço de destino, mas o endereço de origem é falsificado com o endereço da vítima. As respostas a esse pacote são para o endereço da vítima, assim todas as estações da rede respondem para o endereço da vítima, tornando-a inutilizável e até podendo causar falhas no funcionamento da rede. O *Smurf* é considerado um ataque de negação de serviço distribuído.

A varredura de portas consiste em verificar quais são os serviços ativos em um servidor, isto é, se há uma porta TCP escutando requisições e um processo executando para tratar as requisições que chegam a essa porta. No entanto, a varredura de portas é considerada uma ameaça já que metade dos ataques é precedida por uma varredura de portas para identificar as vulnerabilidades do sistema que podem ser exploradas. Na varredura de portas, o atacante gera pacotes e monitora as respostas para determinar se um serviço está vulnerável ou não. **A varredura SYN** é a mais conhecida pela simplicidade e velocidade, e também é conhecida como *half-open scanning*, já que nunca abre uma conexão completa. Um pacote SYN é enviado e, se o sistema está escutando conexões TCP na porta para qual o pacote é destinado, a vítima responderá com um pacote SYN-ACK e, logo, o atacante enviará um RST, fechando a conexão antes que o *handshake* seja estabelecido. Se o serviço não está aberto, a vítima responderá com um pacote RST. Se não há resposta, após várias tentativas, a porta é marcada como filtrada, isto é, há um *firewall* que impede a comunicação entre o atacante e a vítima por tal porta.

A **varredura de conexão TCP**, também conhecido como *full-open scanning*, tem um comportamento semelhante ao anterior, mas neste caso a comunicação é feita totalmente se a porta estiver aberta, ou seja, o atacante envia o SYN, espera pelo SYN-ACK e, por fim, envia o ACK para estabelecer a conexão. Se a porta estiver fechada, o atacante recebe um pacote RST/ACK.

Os ataques de TCP **FIN, XMAS, ACK, NULL** e **Maimon** são semelhantes, já que a ausência de resposta significa que a porta da vítima está aberta ou que existe um *firewall* no caminho entre o atacante e a vítima. No **FIN** é enviado um pacote TCP com o bit FIN ativo em um. Dependendo do sistema operacional da vitima, se a porta da vítima está aberta não terá resposta e se estiver fechada a vítima responderá com um pacote RST/ACK. O **TCP XMAS** simula o comportamento normal de um cliente, já que envia pacotes TCP com os bits URG, PUSH e FIN ativos em um mesmo pacote. De novo, se a porta estiver aberta não haverá resposta e se estiver fechada terá como resposta um pacote RST/ACK. No **NULL**, ao contrário do FIN, é enviado um pacote com todos os bits em zero, sem resposta quando a porta está aberta e com RST/ACK quanto está fechada. A varredura de **TCP Maimon**, nome recebido devido a seu criador, envia um pacote TCP com bit FIN ativo e também com ACK ativo.

A varredura **TCP por Janela** utiliza diferentes tamanhos de janela a uma resposta RST. Dependendo do sistema operacional, as portas abertas usam um tamanho de janela positivo enquanto as fechadas têm uma janela de tamanho zero. Alternativamente aos protocolos de transporte convencionais TCP, UDP e ICMP, é criada a varredura de **SCTP INIT**. Essa técnica tem um comportamento semelhante à varredura TCP SYN, já que cria uma "meia" conexão. Um *chunk* INIT é enviado pelo atacante. Um *chunk* INIT-ACK indica que a porta está aberta,

enquanto um *chunk* ABORT é indicativo de fechada. Se nenhuma resposta for recebida após várias retransmissões, a porta é marcada como filtrada.

3.2. Análise do Conjunto de Dados

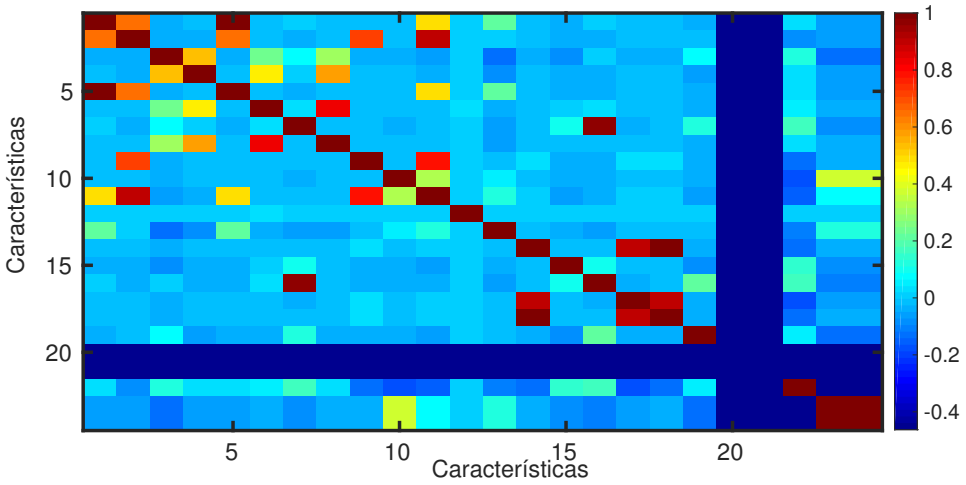


Figura 1: Matriz de correlação das 24 características disponíveis no conjunto de dados. Os pontos escuros em vermelho indicam a máxima correlação e os pontos em azul indicam a correlação mínima.

A Figura 1 mostra a correlação entre as 24 características que compõem o conjunto de dados usado. A matriz calcula a correlação a partir da distância de Pearson (*Pearson Product-Moment Correlation Coefficient* - PPMCC). O produto de Pearson é uma medida da dependência linear entre duas variáveis X e Y, sendo +1 quando a correlação entre as variáveis é linear, 0 quando não tem correlação e -1 quando a correlação entre as variáveis é inversamente linear. Na figura, a correlação linear é representada com vermelho escuro, a correlação nula é representada pela cor branca e o azul representa a correlação inversa. As características 21 e 22 indicam "Quantidade de ECE Flags" e "Quantidade de CWR Flags", respectivamente. As etiquetas de *Explicit Congestion Notification* (ECN) e de *Congestion Window Reduced* (CWR) são usadas para alertar ao remitente de possíveis situações de congestionamento na rede, a fim de evitar a perda de pacotes e retransmissões. Na matriz de correlação, essas duas características estão representadas em azul escuro indicando um baixo nível de correlação. No caso do conjunto de dados criados, as variáveis 21 e 22 não agregam nenhuma informação aos dados, pois são sempre nulas. Isso se deve ao fato que na rede em que os dados foram obtidos não havia congestão ou sobrecarga. Por outro lado, as características 23, "Quantidade de Tipos ICMP", e 24, "Quantidade de Códigos ICMP", são representadas em vermelho. Assim, essas duas variáveis estão altamente correlacionadas entre si, pois o tipo do ICMP está relacionado como o código ICMP.

O conjunto de dados conta com mais de 95 GB em pacotes capturados, resultando em 214.200 fluxos compostos de tráfego malicioso e normal. A Figura 2 mostra a relação entre as classes no conjunto de dados. A classe normal possui aproximadamente 70% dos dados do conjunto, com 106.955 amostras etiquetadas como tráfego normal, a classe de Negação de Serviço representa 10% do conjunto de dados, com 16.741 amostras, e a classe de varredura de portas possui 20% das amostras, ou seja, 30.491 fluxos.

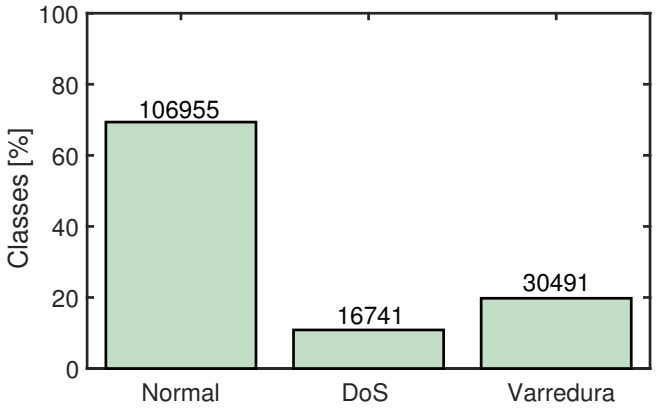


Figura 2: Distribuição das classes no conjunto de dados. A classe principal é a Normal com quase o 70% dos dados, a classe de Negação de Serviço (DoS) tem o 10% e a classe de varredura corresponde ao 20% do conjunto de dados.

4. A Seleção Baseada na Correlação de Características

O método de seleção de características estudado nesse artigo utiliza a correlação entre as variáveis através do coeficiente de Pearson. O valor do coeficiente de Pearson é $-1 \leq \rho \leq 1$, em que 1 significa que duas variáveis estão diretamente correlacionadas através de uma relação linear e -1, o caso inverso, correlação inversa ou anticorrelação.

O coeficiente de Pearson ρ é calculado em termos de média μ e o desvio padrão σ como:

$$\rho(A,B) = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{A_i - \mu_A}{\sigma_A}\right)\left(\frac{B_i - \mu_B}{\sigma_B}\right), \tag{1}$$

ou como função da co-variância σ como:

$$\rho(A,B) = \frac{cov(A,B)}{\sigma_A \sigma_B}. \tag{2}$$

Primeiramente, é preciso obter a matriz de correlação baseada na Equação 2. A matriz de correlação apresenta o cálculo da covariância entre cada par de variáveis. O método de seleção de características estudado neste trabalho atribui um peso w a cada variável. O peso w é uma medida da importância de cada variável que representa uma característica em relação às demais variáveis aleatórias. O peso w para cada característica é dado pela soma do valor absoluto da correlação entre a característica considerada e as demais características do conjunto de dados. A soma é em valor absoluto, pois o coeficiente de Pearson ρ pode apresentar valores negativos, porém, de maneira qualitativa, uma variável que apresenta uma forte correlação inversa com outra aporta tanta informação para a representação dos dados quanto uma variável que tenha o mesmo valor absolto de correlação direta. O valor absoluto da correlação indica o quanto uma variável é linearmente dependente de outra. Correlação próxima a 1 representa uma forte dependência linear. Correlação próxima a 0 representa independência linear entre as duas variáveis. Além disso, o peso w é uma indicação da quantidade de informação que uma variável tem independentemente das outras. O peso w tem valores entre $0 \leq w \leq N$, onde N é a quantidade de características no conjunto de dados, e 0 representa uma variável totalmente independente das outras. Quanto mais alto é o valor do peso w, maior é a dependência linear com outras variáveis e menor é a quantidade de informação que a variável aporta à representação do conjunto de dados.

Algoritmo 1: Seleção de Características Baseada na Correlação.

Input : X: Matriz de Dados
Output: **r**: Vetor de Caraterísticas pontoadas, **w**: Vetor de pesos

```
1 A = Corr(X) /* Calculo da Matriz de Correlação          */
2 for 0 ≤ i < len(A) do
3 |   w_i = 0
4 |   for 0 ≤ j < len(A_i) do
5 |   |   k_i = abs(A_ij) /* Calculo do Valor Absoluto    */
6 |   |   w_i += k_i /* Calculo do Peso                   */
7 |   end
8 end
9 r = sort(w, reverse = True) /* Ordenamento descendente de w */
```

5. Caso de Uso de Seleção de Características: Classificação de Tráfego

Para a classificação automática de tráfego são utilizados os algoritmos mais utilizados de aprendizado de máquina [Buczak e Guven 2015]. Em todos os métodos, o treinamento é realizado com 70% do conjunto de dados e a avaliação com os 30% restante. Durante a fase de treino foi realizada a validação cruzada para evitar a superespecialização ou *overfitting*. Na validação cruzada, o conjunto de dados de treino é dividido em diversas partes e algumas delas não são utilizadas para estimação dos parâmetros. Elas são usadas posteriormente para verificar se o modelo é geral o suficiente para se adaptar a novos dados, evitando a superespecialização.

Na árvore de decisão, as folhas representam a classe final, e os ramos representam as condições baseadas no valor de uma das características de entrada. Durante o processo de treinamento o algoritmo de C4.5 determina a estrutura do tipo da árvore para a classificação. A implementação em tempo real da árvore de decisão consiste em regras "*if-then-else*" que são geradas na forma da árvore previamente calculadas.

As redes neurais são baseadas no cérebro humano no qual cada neurônio realiza uma pequena parte do processamento, transferindo o resultado para o próximo neurônio. Nas redes neurais artificiais, a saída representa o grau de pertinência para cada classe. Os vetores de pesos Θ são calculados durante o treinamento e determinam o peso de cada conexão dos neurônios. No treinamento, há erros causados por cada parâmetro, que são minimizados através do algoritmo de propagação posterior.

Para determinar a qual classe pertence uma amostra, cada camada de rede neural calcula as seguintes equações:

$$z_{(i+1)} = \Theta_{(i)} a_{(i)} \quad (3) \qquad a_{(i+1)} = g(z_{(i+1)}) \quad (4) \qquad g(z) = \frac{1}{1+e^{-z}} \quad (5)$$

onde $a_{(i)}$ é um vetor que determina a saída das camadas i, para a camada $i+1$, e $a_{(i+1)}$ é a saída da camada $i+1$. A função $g(z)$ é a função *Sigmoid* que é a função de ativação de cada neurônio, importante para a classificação. Para altos valores de z, $g(z)$ retorna um e para valores baixos de z, $g(z)$ retorna zero. Logo, a camada de saída retorna o grau de pertinência para cada classe entre zero e um, classificando a amostra como a de maior valor.

As Máquinas de Vetores de Suporte (*Support Vector Machine* - SVM) são um classificador binário baseado no conceito de hiperplanos de separação entre classes que define os limiares da decisão. O algoritmo de SVM classifica mediante a construção de um hiperplano

em um espaço multidimensional que divide as diferentes classes. Um algoritmo iterativo minimiza a função de erro, encontrando o melhor hiperplano de separação. Uma função de *kernel* define o hiperplano, sendo linear ou não linear. Assim, o SVM encontra a margem máxima de separação entre duas classes. A classificação em tempo real é realizada para uma das classes: normal ou não normal; DoS ou não DoS; e varredura ou não varredura. Uma vez que o SVM calcula a saída, a classe é escolhida segundo a maior pontuação. A pontuação de uma amostra x é a distância de x para os limites de decisão que vão desde $-\infty$ a $+\infty$. A pontuação do classificador:

$$f(x) = \sum_{j=1}^{n} \alpha_j y_j G(x_j, x) + b, \tag{6}$$

onde $(\alpha_1, ..., \alpha_n.b)$ são os parâmetros estimado do SVM, e $G(x_j, x)$ é o *kernel* utilizado. Neste trabalho o *kernel* usado foi um *kernel* linear, $G(x_j, x) = x_j^{'}x$, o qual apresenta um bom desempenho com uma mínima quantidade de parâmetros de entrada.

5.1. Resultados

O algoritmo de Seleção de Características desenvolvido foi comparado com a Análise de Componentes Principais (PCA), o algoritmo de ReliefF, a eliminação por Seleção Sequencial para a Frente (*Sequential Forward Selection* - SFS), e a Seleção de Características Recursiva usando SVM (*Support Vector Machine Recursive Feature Elimination* SVM-RFE). Todos os métodos são comparados considerando a seleção de quatro e seis características. A mérito de justiça na comparação, todos os métodos de seleção e redução de características foram executados com os classificadores apresentados anteriormente: árvore de decisão com o algoritmo de CART com 4096 folhas; redes neurais com uma camada escondida e dez neurônios; e máquinas de vetores de suporte (SVM) com *kernel* linear. Todos os experimentos são realizados com validação cruzada 10-*fold*.

A Figura 3 mostra os autovalores associados ao conjunto de dados proposto. É possível ver que com quatro e seis variáveis se obtém entre o 80% e 90% do total da variância. Dessa forma, todas as demais componentes representam entre 10% e 20% da variância e, portanto, podem ser eliminadas sem prejuízos para a representação dos dados.

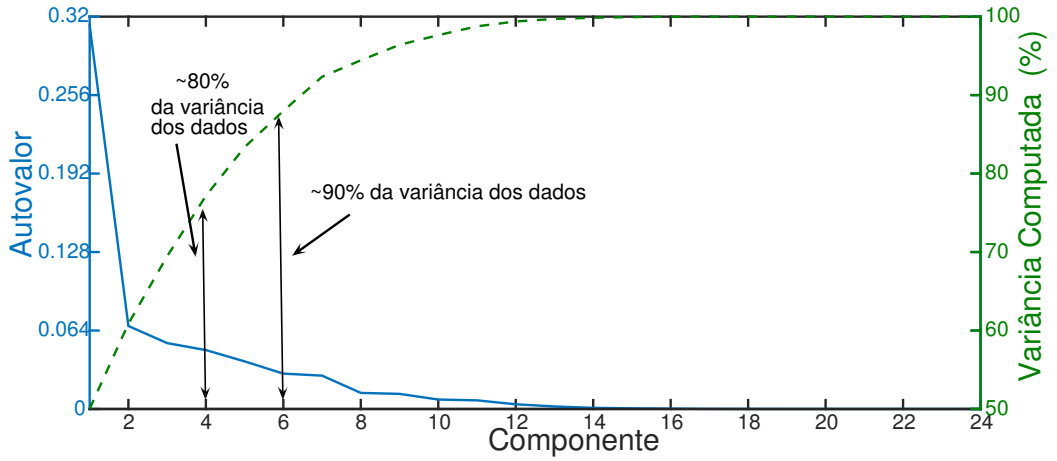


Figura 3: Autovalor para cada uma das 24 características do fluxo. O autovalor associado a cada uma das características transformadas é proporcional à variância dos dados. Entre a quarta e a sexta componente mais importantes, representam de 80% a 90% do total da variância dos dados.

A Figura 4 mostra o desempenho comparando a acurácia dos três classificadores. No primeiro grupo, são mostrados os resultados da árvore de decisão, na qual o algoritmo desenvolvido possui o melhor desempenho com seis características com 97.4% de acurácia. Na sequência, há resultado do PCA com quatro e seis características em 96% e 97.2% respectivamente. O *Sequential Forward Selection* (SFS) apresenta o mesmo resultado com quatro e com seis características com acurácia a 95.5%. O algoritmo de ReliefF também apresenta o mesmo resultado para quatro e seis características, 91.2%. Finalmente o pior resultado é apresentado pelo SVM-RFE com quatro e seis características, 80.2% e 85.8%.

No segundo classificador, a rede neural, o melhor resultado é apresentado pelo PCA com seis características com 97.6% de acurácia. ReliefF apresenta o mesmo resultado com ambas em 90.2%. O algoritmo de seleção por correlação proposto mostra um resultado de 83.9% e 85.0% com quatro e seis características. Por outro lado, o SFS apresenta o pior resultado de todos os classificadores com 78.4% com quatro características e 79.2% com seis. Um resultado inesperado foi o do SVM-RFE, já que com quatro características apresenta um desempenho baixo de 73.6% de acurácia, sendo um dos piores em todos os classificadores, no entanto, com seis características é o segundo melhor para as redes neurais com 90.1%. Vale ressaltar que as características consideradas pelo PCA são variáveis transformadas que podem ser combinações lineares de um maior número características originais, enquanto nos demais métodos são escolhidos subconjuntos de características que limitam o conjunto de dados tratado pelos classificadores.

Nas Máquinas de Vetores de Suporte (SVM) o PCA apresenta um comportamento semelhante ao apresentado nas redes neurais. Com seis características tem a melhor acurácia de todos os classificadores com 98.3%, mas só 87.8% para quatro características. ReliefF novamente apresenta um resultado semelhante para ambas características em 91.4%. O algoritmo proposto tem 84% de acurácia com quatro e 85% com seis características. SFS apresenta a mesma acurácia para as duas amostras 79.5%. Finalmente, o resultado mais baixo para esse classificador é o SVM-RFE com 73.6% em ambos os casos.

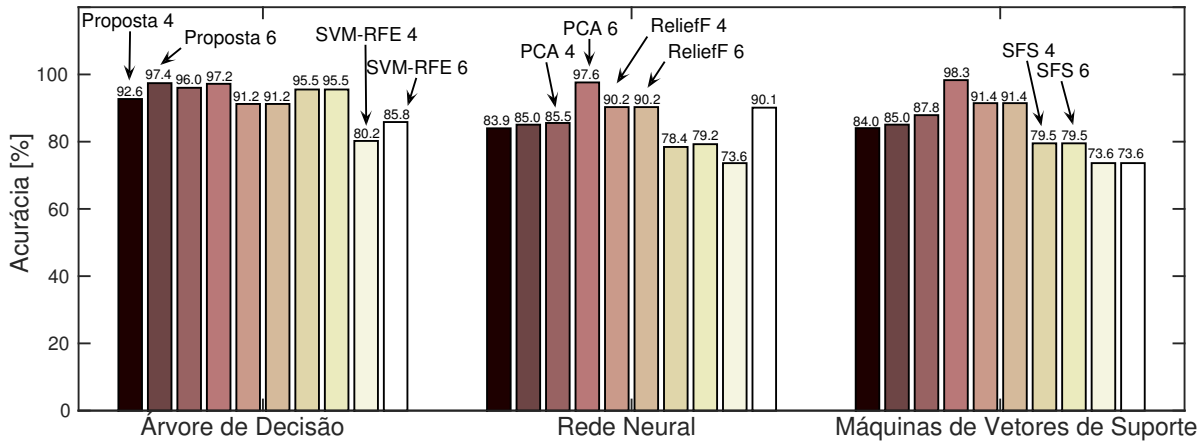


Figura 4: Comparação da acurácia em três classificadores dos métodos de Seleção e redução de características. Nosso algoritmo proposto, PCA Lineal, ReliefF, Sequence Forward Selection (SFS) e SVM-RFE na árvore de decisão, na rede neural e na máquina de vetores de suporte.

Na Tabela 2 é apresentada a seleção feita por cada um dos métodos. Foi excluído o método do PCA já que a saída apresentada é composta por novas características sintéticas e

Tabela 2: Comparação da seleção de características de cada um dos métodos.

	Proposta	ReliefF	SFS	SVM-REF
Número de Característica	12,15,19, 13,10,7	2,11,9, 19,22,3	15,11,8, 5,20,21	14,12,13, 24,23,22
Nome da Característica	qtd_tos, packet_len_m, qtd_rst_flag, ttl_m, qtd_pkt_icmp, qtd_ack_flag	qtd_src_port, qtd_pkt_ip, qtd_pkt_udp, qtd_rst_flag offset_m, qtd_dst_port	packet_len_m, qtd_pkt_ip, qtd_urg_flags, qtd_syn_flag, qtd_ece_flag, qtd_cwr_flag	header_len_m qtd_tos ttl_m qtd_cdg_icmp qtd_t_icmp offset_m

não corresponde com uma comparação dos outros métodos. Todos os métodos mostram as seis características mais importantes. É possível ver que nenhum método escolhe o mesmo grupo de características. No entanto, ReliefF e SFS selecionaram como a segunda melhor característica a número 11, "*qtd_pkt_ip*". Um resultado bem interessante é que o SFS selecionou as características 20 e 21, "*qtd_ece_flag* e *qtd_cwr_flag*". Na matriz de correlação apresentada na Seção3 foi discutida que essas duas características não adicionam nenhuma informação por que são variáveis vazias. Contudo, depois de uma análise foi reparado que a característica mais importante é a número 15 "*pkt_len_m*". Nesse conjunto de dados, a medida do tamanho de pacotes é fundamental para a classificação dos ataques.

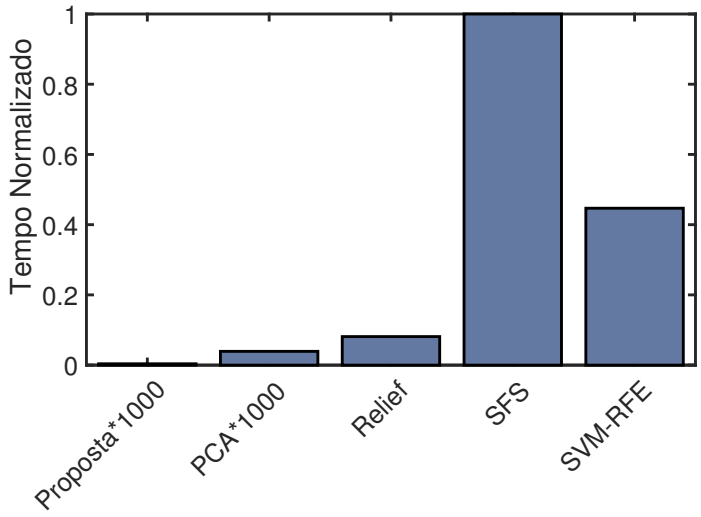


Figura 5: Avaliação de desempenho em relação ao tempo de processamento dos algoritmos de seleção de características. O algoritmo proposto é até 3 vezes mais rápido que o PCA.

Na Figura 5 é apresentado o tempo de processamento de cada um dos algoritmos durante a seleção de características. Os valores são normalizados dividindo o tempo de processamento de cada método pelo valor máximo, obtido pelo SFS. Tanto o algoritmo desenvolvido como o PCA foram multiplicados por 1000 por motivos de escala. O SFS mostra o pior desempenho, porque apresenta a maior complexidade, com tempo médio de processamento de 83.480 segundos. O SFS é um algoritmo que realiza múltiplas iterações a fim de minimizar o erro quadrático médio. Consequentemente, todas as interações incrementam o tempo de processamento. O algoritmo de seleção de características desenvolvido nesse trabalho mostra o menor

tempo de processamento, aproximadamente 0.72 segundos em média, seguido do PCA, com 2.04 segundos em média. Os dois métodos apresentam um bom despenho no cálculo de tempo de processamento, já que ambos utilizam a multiplicação de matrizes para obter o conjunto de características. A multiplicação de matrizes é uma função computacional simples já que pode ser paralelizada. As medidas foram obtidas e um computador com processador Intel Xeon com frequência de relógio de 2.6 GHz e 256 GB de memória RAM.

6. Conclusão

Esse artigo apresentou um algoritmo de seleção de características para a classificação de tráfego de redes. O algoritmo calcula a correlação das variáveis de modo não supervisionado. A fim de realizar uma avaliação, os resultados obtidos a partir do algoritmo apresentado de seleção de características foram comparados com os obtidos através métodos populares de seleção e redução de características. Os métodos de seleção comparados são o ReliefF, como método de filtragem, a Seleção Sequencial de Características (*Sequential Feature Selection* - SFS), como método *wrapper*, a Seleção Recursiva de Eliminação usando Máquinas de Vetores de Suporte (*Support Vector Machine Recursive Feature Elimination* - SVM-RFE), como método embarcado, e finalmente a Análise de Componentes Principais (*Principal Component Analises* - PCA), como método de projeção de características. Para a avaliação e comparação dos métodos de seleção, foi utilizado um conjunto de dados de segurança que apresenta mais 16 tipos diferentes de ataques assim como tráfego normal. Nesse conjunto de dados foram aplicados três algoritmos de classificação como a árvore de decisão, redes neurais e as máquinas de vetores de suporte. O algoritmo desenvolvido de filtragem apresentou uma acurácia de 97.4% com 6 características selecionadas no algoritmo de árvore de decisão. Além disso, o algoritmo apresentado reduz o tempo de processamento em mais de 100 vezes quando comparado com o método de pior desempenho.

Referências

Ang, J. C., Mirzal, A., Haron, H. e Hamed, H. N. A. (2016). Supervised, Unsupervised, and Semi-Supervised Feature Selection: A Review on Gene Selection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 13(5):971–989.

Bar, A., Finamore, A., Casas, P., Golab, L. e Mellia, M. (2014). Large-scale network traffic monitoring with DBStream, a system for rolling big data analysis. Em *2014 IEEE International Conference on Big Data (Big Data)*, páginas 165–170. IEEE.

Buczak, A. e Guven, E. (2015). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys Tutorials*, (99):1–26.

Chandrashekar, G. e Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28.

Guyon, I., Weston, J., Barnhill, S. e Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422.

Hall, M. A. (1999). *Correlation-based Feature Selection for Machine Learning*. PhD thesis, The University of Waikato.

Heidemann, J. e Papdopoulos, C. (2009). Uses and challenges for network datasets. Em *Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications & Technology*, páginas 73–82. IEEE.

Hu, P., Li, H., Fu, H., Cansever, D. e Mohapatra, P. (2015). Dynamic defense strategy against advanced persistent threat with insiders. Em *IEEE Conference on Computer Communications (INFOCOM)*, páginas 747–755.

Lee, W., Stolfo, S. J. e Mok, K. W. (1999). Mining in a data-flow environment: Experience in network intrusion detection. Em *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 114–124. ACM.

Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., Weber, D., Webster, S. E., Wyschogrod, D., Cunningham, R. K. e others (2000). Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. Em *Proceedings of DARPA Information Survivability Conference and Exposition. DISCEX'00.*, volume 2, páginas 12–26. IEEE.

Mayhew, M., Atighetchi, M., Adler, A. e Greenstadt, R. (2015). Use of machine learning in big data analytics for insider threat detection. Em *IEEE Military Communications Conference, MILCOM*, páginas 915–922.

Mladenić, D. (2006). Feature Selection for Dimensionality Reduction. Em Saunders, C., Grobelnik, M., Gunn, S. e Shawe-Taylor, J., editors, *Subspace, Latent Structure and Feature Selection (SLSFS): Statistical and Optimization Perspectives Workshop.*, páginas 84–102. Springer Berlin Heidelberg, Bohinj, Slovenia.

Pastana Lobato, A. G., Andreoni Lopez, M. e Duarte, O. C. M. B. (2016). Um Sistema Acurado de Detecção de Ameaças em Tempo Real por Processamento de Fluxos. Em *SBRC'2016*, páginas 572–585, Salvador, Bahia.

Paxson, V. (1999). Bro: a System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23-24):2435–2463.

Robnik-Šikonja, M. e Kononenko, I. (2003). Theoretical and Empirical Analysis of ReliefF and RReliefF. *Machine Learning*, 53(1/2):23–69.

Roesch, M. (1999). Snort-Lightweight Intrusion Detection for Networks. Em *Proceedings of the 13th USENIX conference on System administration*, páginas 229–238. USENIX Association.

Schölkopf, B., Smola, A. J. e Müller, K.-R. (1999). Kernel principal component analysis. Em *Advances in kernel methods*, páginas 327–352. MIT Press.

Sommer, R. e Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. Em *IEEE Symposium on Security and Privacy (SP)*, páginas 305–316. IEEE.

Stonebraker, M., Çetintemel, U. e Zdonik, S. (2005). The 8 requirements of real-time stream processing. *ACM SIGMOD Record*, 34(4):42–47.

Tavallaee, M., Bagheri, E., Lu, W. e Ghorbani, A.-A. (2009). A detailed analysis of the KDD CUP 99 data set. Em *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications*. IEEE.

Vallentin, M., Sommer, R., Lee, J., Leres, C., Paxson, V. e Tierney, B. (2007). The NIDS Cluster: Scalable, Stateful Network Intrusion Detection on Commodity Hardware. Em *Recent Advances in Intrusion Detection*, páginas 107–126. Springer Berlin Heidelberg, Berlin, Heidelberg.

Van Der Maaten, L., Postma, E. e den Herik, J. (2009). Dimensionality reduction: a comparative. *Journal of Machine Learning Research*, 10:66–71.

Quantifying Node Security in Wireless Sensor Networks under Worm Attacks

Alex Ramos[1], **Breno Aquino**[1], **Raimir Holanda Filho**[1], **Joel J. P. C. Rodrigues**[1,2,3]

[1]Graduate Program in Applied Computer Science (PPGIA)
University of Fortaleza (UNIFOR) – CE – Brazil

[2]National Institute of Telecommunications (Inatel) – MG – Brazil

[3]Instituto de Telecomunicações, Universidade da Beira Interior, Portugal

{alex.lacerda, brenoaquino}@edu.unifor.br,

raimir@unifor.br, joeljr@ieee.org

Abstract. *The peculiar characteristics of wireless sensor networks (WSNs) make them vulnerable to physical attacks. Once a sensor node is physically captured by an adversary, it can be modified not only to perform malicious activities to disrupt network operation but also to propagate malicious worms to infect other nodes. In the face of such a threatening scenario, the system administrator needs to be aware of which nodes may have been compromised, so that appropriate countermeasures can be taken in a timely fashion. This paper presents the Sensor Security Status (S3), a security metric model for estimating in an online manner the probability that a sensor node has been infected, based on both the interaction among nodes and the alerts from the intrusion detection system (IDS). Simulation results show that S3 can accurately estimate node security level with low performance overhead and power consumption.*

1. Introduction

The situational awareness provided by security metrics is essential to help system administrators take informed decisions regarding the security status of a network and its components [Zonouz et al. 2015]. In wireless sensor networks (WSNs), a metric that is able to quantify the security level of sensor nodes can be used, for example, to identify which nodes should be trusted (to provide reliable data) or which nodes need careful attention from attack response mechanisms. Ultimately, the security level of nodes can be used to determine the security status of the entire WSN and the sensor data it provides for users [Ramos and Filho 2015].

Although several security metrics have been proposed for traditional networks [Pamula et al. 2006, Wang et al. 2007], the unique characteristics of sensor nodes make it impossible to directly apply those metrics to WSNs. This requires security metrics especially designed to quantify security based on the specific attack types and vulnerabilities of sensor networks [Ramos and Filho 2015]. In particular, the resource constraints, and deployment in open areas make sensor nodes vulnerable to physical attacks. In this type of attack, an adversary captures a node and retrieves secret information from its memory in order to gain access to the network. In addition, the adversary can compromise the captured node to make it perform several malicious activities in the WSN (e.g., routing attacks like sinkhole, misdirection, etc) [Walters et al. 2007].

To easily gain full control of the network, the adversary may also attempt to have the captured node propagate malicious worms to compromise more nodes via wireless communication. Once a node is infected by the worm, it will be able to both behave maliciously (*e.g.,* carry out routing attacks) and re-propagate the worm to infect other nodes [Francillon and Castelluccia 2008, Haghighi et al. 2016]. This makes worms one of the most devastating types of attack.

To detect misbehaving nodes, an intrusion detection system (IDS) can be deployed in the WSN [Raza et al. 2013]. Although IDSs are usually not capable of identifying all intrusions that occur (because the dynamic operation of WSNs may make it difficult to distinguish normal behavior from malicious behavior), the malicious nodes that the IDS is actually able to identify indicate that the network is under attack and, consequently, that other nodes may also have been compromised.

Therefore, to effectively portray the current security status of nodes and provide administrators with useful information when attacks are occurring, a security metric for WSNs should consider two main factors: (1) benign nodes can be compromised by malicious nodes (*i.e.,* worm attacks); and (2) to avoid that some compromised nodes go undetected, the intrusions that the IDS is able to identify can be used to predict other intrusions. Intuitively, the greater the number of malicious nodes the IDS detects, the higher the chance that other nodes have also been compromised.

The few existing security metrics for WSNs ([Anand et al. 2005, Ramos and Filho 2015]) fail to handle these aspects, because they either ignore IDS alerts or disregard the fact that a node can be compromised by other nodes. To address those limitations, this paper presents a security metric model called *sensor security status (S3)*. The S3 model uses IDS alarms received in real-time to estimate how much the security level of each sensor node of a WSN has been affected by intrusions. This assessment is performed using an *attack propagation graph (APG)*. Considering that adversaries can take advantage of the network communication pattern to propagate worm attacks [Ho 2015], the APG captures how nodes can compromise others through their communication behavior.

The attack propagation graph is automatically constructed during an initial configuration phase when sensor nodes behave normally. Then, the APG is transformed into a Bayesian network (BN) so that inferences about the security status of nodes can be made. More precisely, when a new alert is raised by the IDS, a belief propagation algorithm, the Gibbs sampler [Casella and George 1992], is applied to compute the probability that each WSN node has been affected by the intrusion and has become compromised.

The remainder of this paper is organized as follows. Section 2 presents system models and assumptions. Section 3 describes the proposed S3 model. Section 4 provides an evaluation of S3. The past related work is reviewed in Section 5. Finally, Section 6 concludes this paper.

2. System Model and Assumptions

This section presents network, attack, and security models, as well as other assumptions considered.

2.1. Network Model

The base station (BS) is assumed to be a central command node with no resource constraint problem and it cannot be compromised by attacks. it is also assumed that the WSN

is comprised of static nodes that periodically send sensor readings to the base station by means of a multi-path routing protocol such as the standardized Routing Protocol for Low Power and Lossy Networks (RPL) [Winter et al. 2012]. In RPL, a destination-oriented directed acyclic graph (DODAG) is created to enable message forwarding from sensor nodes to the DODAG root (*i.e.,* the base station). Each node knows its RPL-parents but has no information regarding its children. Every node periodically chooses a *preferred* PRL-parent to forward its messages, as shown in the left hand side of Fig. 1. The preferred RPL-parent of each node is chosen from the parent set and is periodically updated according to some predefined routing metrics (*e.g.,* remaining energy, link quality).

In the physical and link layers, sensor nodes are assumed to implement the IEEE 802.15.4 protocol which is the *de facto* standard for low power and lossy wireless networks such as WSNs.

2.2. Attack Model

WSNs can be target of several types of attack [Walters et al. 2007]. S3 assumes that attacks can be initiated from node capture. A node that is physically captured by an adversary can perform malicious activities to disrupt network operation. Those activities include attacks such as selective forwarding, sinkhole, and data alteration [Raza et al. 2013]. In addition, a compromised node can transfer data packets with malicious code to compromise its neighbors (worm attack) [Francillon and Castelluccia 2008, Haghighi et al. 2016]. This worm propagation process can repeat itself and lead to the compromise of the whole network if countermeasures are not taken [Ho 2015]. Therefore, an adversary can compromise an entire WSN with a single node capture. It is important to note that researchers have developed practical worm attacks on both Harvard architecture (*e.g.,* Mica motes) [Francillon and Castelluccia 2008] and Von Neumann architecture (*e.g.,* TelosB) sensor devices [Giannetsos et al. 2009].

To attain their objective of maximizing the amount of compromised nodes, worms can apply different propagation strategies. Although broadcasting a worm to all neighbors may seem to be the best strategy, it can result in severe congestion of network traffic and hence decrease, or even stop, the propagation rate [Khayam and Radha 2005]. Therefore, we assume an intelligent worm, which seeks to maximize the number of infected nodes by using the normal communication pattern of the network to propagate itself, as discussed in [Ho 2015]. In a RPL-based WSN, this could be accomplished by making compromised nodes transmit the worm only to the current preferred RPL-parent. This strategy is also useful to avoid the worm propagation from being easily identified by any worm detection mechanism that could eventually be present in the network [Ho 2015].

2.3. Security Model

The S3 model assumes the existence of an IDS on the WSN. IDSs such as SVELTE [Raza et al. 2013], for example, can detect compromised nodes that perform malicious activities (*e.g.,* sinkhole, data alteration, etc.). A worm detection mechanism, such as the scheme proposed in [Ho 2015], may be present in the network as well. S3 also assumes the existence of response mechanisms that may attempt to recover malicious nodes detected by the IDS. Node recovery might be achieved in various different ways. A simple recovery approach would be to reload the node's program [De et al. 2009].

3. Sensor Security Status Metric

The goal of S3 is to automatically evaluate the security level of each sensor node in an online manner so as to provide awareness of how secure the network is. The intuition leveraged by S3 is that when a malicious node is identified by the IDS, it is possible that this node has been infected by a worm that have already managed to infect other nodes. Therefore, by using IDS alerts as evidence that an attacker is present in the WSN, S3 attempts to predict future attacks or attacks that may have already occurred but have not been detected by the IDS (yet).

To do so, S3 uses an attack propagation graph to assess how a worm could take advantage of the communication pattern of nodes to spread itself through the network. The APG, which is automatically built during an initial configuration phase, captures the communication dependencies among neighboring sensor nodes. After this phase, the APG is then turned into a Bayesian network that is combined with real-time IDS alerts by means of a node compromise dissemination analysis procedure to estimate the probabilities that sensor nodes have been affected by an attacker.

More specifically, the security measure provided by S3 for each node indicates the probability that the node has been compromised by a malicious worm given that one or more compromised nodes have been detected by the IDS. Accordingly, the security measure provided by S3 is a real value lying between 0 and 1 (inclusive). A larger S3 value indicates less security. Every time the current system state changes, *i.e.,* when a new alert is raised by the IDS or a malicious node is recovered, then the node security measures provided by S3 are updated. In the following, the formalism and mathematical models used to compute S3 are presented.

3.1. Attack Propagation Graph

The APG is a directed acyclic graph which captures all possible paths that sensor nodes can use to forward messages to the base station. Each vertex in the APG represents a sensor node and a direct communication dependency between two nodes exists if data packets (messages) flow from one node to the other, in the direction of the base station. This relationship is represented in the APG by an arrow (directed edge) between the two nodes. For example, if node n_j receives sensor data from node n_i, it is said that node n_j is dependent on n_i, which is represented as $n_i \rightarrow n_j$. Each arrow is labeled with a probability value $Pr(n_i \rightarrow n_j)$ that indicates the fraction of times node n_i forwards messages to the base station through n_j.

The APG of a sample RPL-based WSN is shown in the right hand side of Fig. 1. The probabilities on arrows represent $Pr(n_i \rightarrow n_j)$ values. For example, node E forwards sensor data to the base station through nodes A and B with probabilities 0.85 and 0.1, respectively. In other words, A was the preferred RPL-parent of E 85% of the time, while 10% of the time B was the preferred RPL-parent.

Notice that the APG only represents the fraction of messages successfully delivered to next-hop neighbors. This means that, due to packet loss, the $Pr(n \rightarrow .)$ values of a given node n may not sum up to 1. For example, while 0.95 of E's messages have been successfully delivered to nodes A and B, 0.05 have been lost (*i.e.,* $1 - (0.85 + 0.1)$). This allows the APG to capture the lossy behavior of WSN's communication links.

Generation of the APG: To obtain the information required to build the APG, an information collection agent (ICA) is installed in each sensor node. During the initial

configuration phase, this agent maintains a variable that counts the number of messages the node has successfully delivered to each of its parents. To do so, the agent periodically sends dummy messages to the current preferred RPL-parent of the underlying node. Since RPL periodically chooses a new preferred parent from the parent set of each node, when the configuration phase terminates, the counters stored by the agent will correspond to the number of times each parent was chosen as the preferred parent.

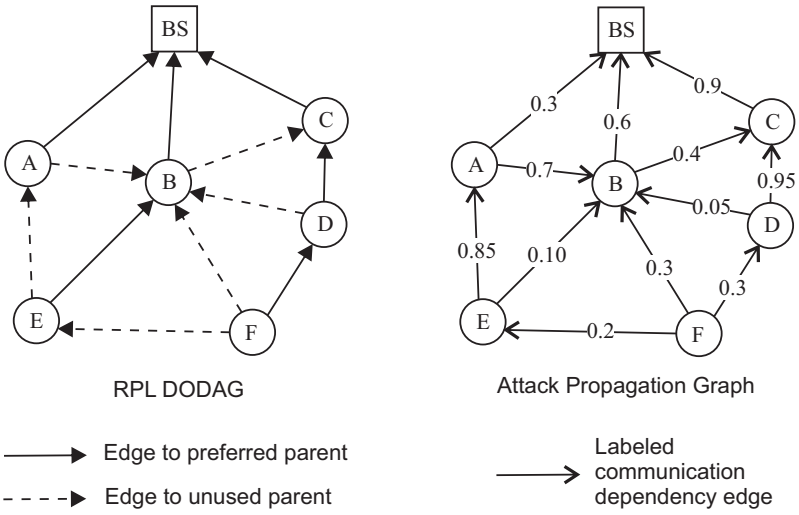


Figure 1. RPL DODAG and corresponding APG of sample WSN.

The number of messages lost is also stored. To accomplish this, ICA interacts with the link layer protocol. Since the IEEE 802.15.4 is an acknowledgment-based protocol, ICA knows that the node's messages have been successfully delivered to the preferred parent if an ACK frame is received by the link layer. On the other hand, if an ACK is not received by the sender after a predefined threshold time (and a specific number of retransmissions), the link layer considers that the message has been lost. In this case, the lost messages counter of the sender is incremented by ICA.

In the end of the configuration phase, each sensor node sends its counters to the base station along with their associated node IDs. Specifically, a *counter message* sent to the base station by a given node A contains the ID of A, the IDs of A's RPL-parents and their respective (message delivery) counters collected by ICA. The counter message also contains the number of lost messages.

As soon as the counter messages of nodes arrive in the base station, they are parsed in order to generate a *frequency DAG*, which is similar to the APG but with message counters in the edge labels, rather than probabilities. When all counter messages are received, the resulting frequency DAG is then converted into an APG. This is accomplished by converting the frequency labels into probabilities, which is done by dividing each frequency label by the total number of messages sent by the source node (i.e., the total successfully delivered messages + the number of lost messages).

It should be highlighted that ICA only runs on sensor nodes during the initial configuration phase. Moreover, all other steps performed by S3 are performed in the base station, namely, frequency DAG generation, APG generation, BN generation, and BN inference. Furthermore, the number of dummy messages sent by ICA can be decreased

since ICA can take advantage of the sensor readings that are periodically sent by the nodes to the base station (as discussed in Section 2). Depending on the frequency those sensor readings are sent, ICA can maintain its counters updated even if no dummy messages are sent. Therefore, the overhead added by ICA to the WSN is as low as possible.

3.2. Bayesian Network

To model how worms can propagate by taking advantage of the communication pattern of sensor nodes, the APG is translated into a Bayesian network (BN) that captures the probabilities that each node can be compromised by other nodes. More precisely, each APG vertex is modeled as a Bernoulli random variable representing the security state of a node, *i.e.,* 1 (True) if the node is compromised, or 0 (False) otherwise. Since it is assumed that worms propagate according to the network communication behavior, each arrow will represent a cause-consequence relationship between two nodes, meaning that one node can be compromised by the other. Each arrow probability $Pr(n_i \rightarrow n_j)$ will then correspond to the probability that node n_j gets compromised by a worm sent by n_i (in the case that n_i has been directly or indirectly compromised by an attacker). For example, if n_i is a compromised node and $Pr(n_i \rightarrow n_j) = 0.85$, then n_j has 0.85 chance of being compromised by n_i since this number represents the probability that n_j receives a message (containing worms) from n_i (*i.e.,* , the probability that n_j is the preferred RPL-parent of n_i).

Since each node in the BN is directly compromised by its parent nodes (source of incoming arrows[1]), a conditional probability table (CPT) is created (with the aid of arrow probability values) and associated with each node. The CPT in a given node n stores the probability that this node gets compromised (or not) given different combination of states of its BN-parent nodes $Pa[n]$. In other words, the CPT corresponds to the conditional probability distribution $Pr(n|Pa[n])$. Formally, let $Pr(n) = 1 - Pr(\overline{n})$. For $p_n^i \in Pa[n]$, let a_i be the communication arrow $p_n^i \rightarrow n$. Considering that a node cannot be compromised by a worm if none of its BN-parents is compromised, then $Pr(n|Pa[n])$ is defined as follows:

$$Pr(n|Pa[n]) = \begin{cases} 0, & \forall p_n^i \in Pa[n], p_n^i = 0, \\ Pr\left(\bigcup_{p_n^i=1} a_i\right), & otherwise. \end{cases} \quad (1)$$

Considering that a node can become compromised by any of the BN-parents that is already compromised, then the probability $Pr\big(\bigcup_{p_n^i=1} a_i\big)$ is derived as follows:

$$Pr\left(\bigcup_{p_n^i=1} a_i\right) = 1 - \prod_{p_n^i=1} [1 - Pr(a_i)] \quad (2)$$

Fig. 2 illustrates how a CPT is generated for a sample BN. For example, node B cannot become compromised if none of its BN-parents is compromised, *i.e.,* $Pr(B|\overline{A}, \overline{C}) = 0$. If only the node A is compromised, B gets compromised only

[1]Notice that a RPL-parent node is the opposite of a BN-parent node. For example, for an edge $n_i \rightarrow n_j$, node n_j is the RPL parent of n_i, while n_i is the BN-parent of n_j.

when messages are received from A, *i.e.,* $Pr(B|A,\overline{C}) = 0.6$. If both BN-parents are compromised, B will become compromised when messages are received by either of its BN-parents, *i.e.,* $Pr(B|A,C) = 1-(1-0.6)\times(1-0.9) = 0.96$.

Note that since node A has no parents, its prior probability is set to a value very close to zero (*i.e.,* 0.0001). Alternatively, to account for other uncertainties (*e.g.,* node capture), the administrator could choose a prior probability value that represents his/her subjective belief on the likelihood that node A can be directly compromised by an adversary (rather than a parent node). This uncertainty could be extended to other nodes by changing the zero probability value in Eq. 1.

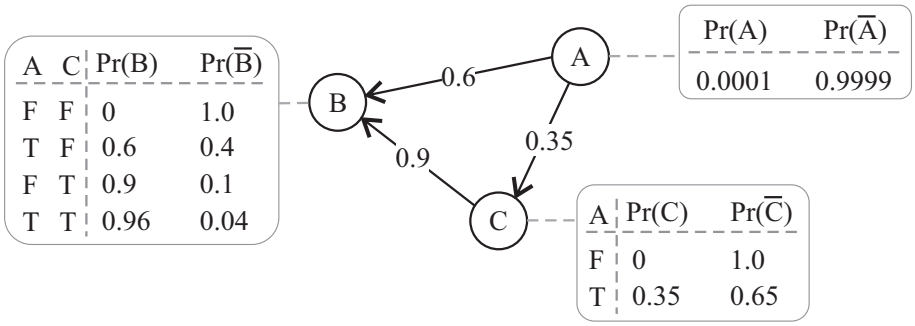


Figure 2. Bayesian network illustration.

3.3. Node Security Level Computation

When the initial configuration phase is concluded, the information collection agents are deactivated and the Bayesian network is generated from the APG in order to estimate the security level of sensor nodes in an online manner. Every time the current system condition changes (*i.e.,* when intrusions are observed by the IDS or compromised nodes are recovered), the state of each vertex in the BN is set accordingly. Then, Bayesian inference techniques of forward and backward propagation are used to update the probability that each sensor node is directly or indirectly affected by the compromised nodes.

Formally, let $N = \{n_1, ..., n_q\}$ be the set of vertices in the BN and $E = \{n'_1, ..., n'_r\} \subset N$ be the set of compromised nodes detected by the IDS (observed attack evidences). Notice that the state of each node in E is true, *i.e.,* $\forall n'_i \in E, n'_i = 1$. Let $n_j \in N - E$ be a node whose posterior probability has to be obtained (*i.e.,* a query node). We are interested in computing the posterior probability of n_j given E, *i.e.,* the conditional probability $Pr(n_j|E)$, which is given as follows:

$$Pr(n_j|E) = \frac{Pr(n_j, E)}{Pr(E)} = \frac{Pr(n_j, n'_1, ..., n'_r)}{Pr(n'_1, ..., n'_r)} \tag{3}$$

Let $H = \{n''_1, ..., n''_k\} \subset N$ be the set of nodes in the BN which are different from query nodes and evidence nodes (*i.e.,* hidden nodes), thus $N = \{n_j\} \cup E \cup H$. The numerator and denominator in Eq. 3 can be expressed using the joint probability of all BN nodes as follows:

$$Pr(n_j|E) = \frac{\sum_{n''_1,...,n''_k \in \{0,1\}} Pr(n_j, n'_1, ..., n'_r, n''_1, ..., n''_k)}{\sum_{n_j,n''_1,...,n''_k \in \{0,1\}} Pr(n'_1, ..., n'_r, n_j, n''_1, ..., n''_k)} \quad (4)$$

In a BN, the joint probability of all vertices is given by the chain rule as:

$$Pr(n_1, ..., n_q) = \prod_{j=1}^{q} Pr(n_j|Pa[n_j]) \quad (5)$$

By combining Eqs. 4 and 5, the posterior probability $Pr(n_j|E)$ can be solved for any node n_j. For example, in Fig. 2, suppose that nodes A and B are identified by the IDS as malicious. The posterior probability of C being compromised is calculated as follows:

$$\begin{aligned} Pr(C|A,B) &= Pr(C,A,B)/Pr(A,B) \\ &= 0.46 \text{ where,} \\ Pr(C,A,B) &= 0.35 \cdot 0.0001 \cdot 0.96 \\ &= 0.0000336, \\ Pr(A,B) &= \sum_{C \in \{0,1\}} Pr(A,B,C) \\ &= (0.0001 \cdot 0.6 \cdot 0.65)_0 + (0.0001 \cdot 0.96 \cdot 0.35)_1 \\ &= 0.0000726 \end{aligned}$$

Since exact inference calculation procedures like the one presented above can become computationally infeasible for large BNs, s3 makes use of an approximate Monte Carlo inference algorithm, namely, the Gibbs sampler. In summary, the Gibbs sampler generates a sequence of samples from a joint probability distribution of a set of random variables $X = \{X_1, ..., X_n\}$. By using a large number of samples, it is possible to approximate the right joint distribution. Specifically, to compute a joint distribution $Pr(X = X_1, ..., X_n|e_1, ..., e_m)$, where e_i is an evidence, the Gibbs sampler initializes X to an arbitrary value in its state space and then samples an adjacent state, with the conditional probability $Pr(X|e)$ conducting the sampling procedure. Repeating the sampling procedure at sufficiently long intervals makes the joint distribution converge.

4. Performance Evaluation

In this section, a simulation-based evaluation of S3 is presented, in terms of its performance and accuracy. The experiments performed allowed to determine: (a) the minimum number of dummy messages that ICA needs to send during the configuration phase; (b) the energy overhead generated by ICA in the WSN; (c) the amount of time required by S3 to both build its fundamental data structures (i.e., APG and BN) and estimate the security metric value of sensor nodes; and finally, (d) the mean error of the estimated metric values.

4.1. Implementation and Experimental Setup

The information collection agent (ICA) is implemented in the Contiki OS [Dunkels et al. 2004], an open source and widely used operating system for WSNs and the Internet of things. Contiki uses extensively tested implementations of both IEEE 802.15.4 and RPL (contikiRPL). The RPL implementation is based on IPv6. Hence, uIP, an IP stack implementation in Contiki, is used to enable IP communication in the WSN.

To implement the approximate inference in the Bayesian network, the DlibC++ [King 2002] open source library is used. This library is widely adopted in both industry and academia.

The experiments were carried out in Cooja [Osterlind et al. 2006], the Contiki network simulator, which has demonstrated to generate realistic results [Raza et al. 2013]. Sensor nodes in Cooja run deployable code and are emulated at the hardware level. In the simulated WSNs, Tmote Sky [Polastre et al. 2005] nodes were used. The base station used was a real laptop that communicated with Cooja by means of a serial socket interface. The laptop was running Ubuntu 16.04 and had a 2.20 GHz Intel Core i5-5200U CPU and 4.0 GB of RAM. Each simulation scenario was run 10 times, and the average and the standard deviation of the results were computed to show their precision.

4.2. Minimum Number of ICA's Dummy Messages

Before estimating the security level of sensor nodes, S3 needs to capture the network communication pattern in the configuration phase and translate it into an APG. To do so, the number of messages sent by ICA should be large enough to build a model that correctly represents the actual network behavior. At the same time, to avoid adding too much overhead in the sensor nodes, the amount of ICA messages sent should be as small as possible.

In this section, the number of dummy messages required for the convergence of the APG parameters is evaluated for a typical WSN consisting of 30 nodes. In each sensor node, the information collection agent sends a total of $3,000$ dummy messages to the current preferred RPL-parent in order to update the counter variables. Therefore, a total of $90,000$ one-hop dummy messages are sent in the network.

In the described scenario, the first parameter evaluated is the number of edges in the APG. Notice that each time a RPL-parent is firstly chosen as preferred by a given node, a new edge should be added in the APG. Fig. 3 shows the APG size *vs.* the number of dummy messages transmitted. The number of edges quickly approaches 175, before $20,000$ messages are sent. Then, it starts stabilizing at approximately $30,000$ messages, i.e., when $1,000$ messages have been transmitted per node.

The second parameter analyzed is the convergence of the probability values (labels) of the APG edges. Specifically, the normalized edge probability updates (i.e., the absolute difference between the current and the updated values) are computed. Fig. 4 shows the average behavior of those values as each node sends its $3,000$ dummy messages. As can be seen, the normalized updates quickly converge to zero at approximately $1,000$ messages. This means that on average the edge probability values begin to converge when each node transmits $1,000$ messages.

It should be highlighted that the convergence of the APG parameters is heavily influenced by the way RPL selects the preferred parents, which in turn depends on the

objective functions and routing metrics configured by the network administrator. However, the experiments provided in this section have shown that when the default behavior of contikiRPL is used, $1,000$ dummy messages sent per node seems sufficient to generate an APG with suitable coverage.

In the simulated scenario, the time interval between two consecutive dummy messages was 35 seconds. Note that this value was used in the simulations but it is not the requirement for ICA. However, reasonably large interval values (like 35s) allow ICA to take advantage of sensor reading messages that nodes periodically send to the base station (usually around one per minute). Hence, ICA can reduce the number of dummy messages sent, which decreases the energy overhead it generates. On the other hand, large intervals have the disadvantage of increasing the duration of the training phase. Therefore, the time interval between dummy messages (i.e., the interval ICA updates its counters) should be chosen taking into account that a large value extends the configuration phase but saves energy, while a small value expends more energy but shortens the configuration phase. For example, if each node sends one dummy message per second, then the time required for the configuration phase would be approximately 17 minutes only.

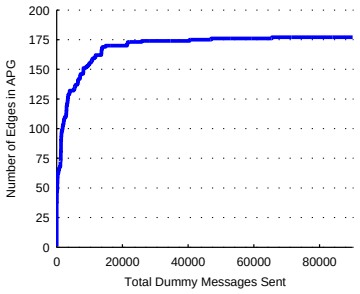


Figure 3. APG's number of edges convergence.

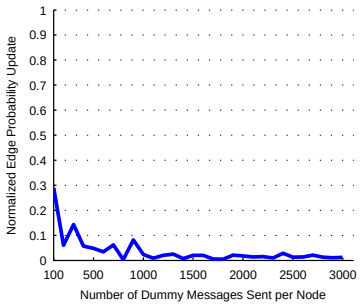


Figure 4. APG's edge probabilities convergence.

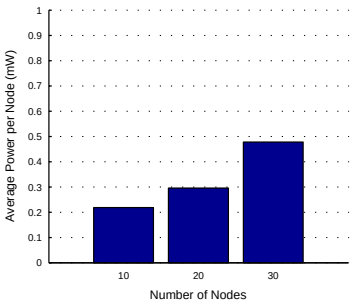


Figure 5. Node's average power consumption.

4.3. Energy Overhead in the WSN

Considering that WSN nodes are battery powered, this section evaluates the amount of power each information collection agent consumes to send $1,000$ dummy messages to its RPL-parents, maintain the counter variables, as well as send the counter messages to the base station in the end of the configuration phase.

In order to do so, the Powertrace [Dunkels et al. 2011] application provided by Contiki is used to measure the power consumption of different operation modes of sensors in terms of the number of clock ticks. The four typical operation modes are: low power mode, or LPM (MCU idle, radio off); CPU mode (MCU on, radio off); listen mode (MCU on, radio receiving); and transmit mode (MCU on, radio transmitting). The power consumption of a node is calculated as follows:

$$\begin{aligned} Power(mW) = \; & (transmit \times 19.5\,mA + listen \times 21.8\,mA + CPU \times 1.8\,mA \\ & + LPM \times 0.0545\,mA) \times 3\,V \;/\; (32768 \times Time(s)) \end{aligned} \tag{6}$$

Where $32,768$ is the number of clock ticks per second of Tmote Sky nodes, $Time(s)$ is the duration of the simulation (in seconds), and the current and voltage values (in mA and in V, respectively) have been obtained from the datasheet of Tmote Sky.

Fig. 5 shows the average power consumption per node (in mW) for three different sized networks, containing 10, 20, and 30 nodes, respectively. As demonstrated, the power consumed by ICA is $0.22\ mW$ in the 10-node WSN, while in the 30-node WSN ICA consumes $0.48\ mW$. Since the power values shown in Fig. 5 are only consumed during the configuration phase, not throughout the entire lifetime of the WSN, it can be concluded that the energy overhead added by ICA is fairly low.

Notice that those results reflect the worst case scenario, in which all $1,000$ dummy messages need to be sent. However, as discussed in the previous section, the time interval ICA updates its counters can be chosen so as to reduce or even eliminate the necessity of transmitting dummy messages.

4.4. Time required to estimate security metric values

As soon as the configuration phase is complete and the frequency DAG is generated, S3 performs three more steps (in the BS): (1) to convert the frequency DAG into an APG; (2) to translate the APG to a Bayesian network; and (3) to run Gibbs sampler on the BN to estimate the security level of all sensor nodes. Note that the first two steps only need to be executed once, while the third step is executed every time new IDS alerts are raised.

In this experiment, four different sized WSNs have been evaluated. The table in Fig. 6 shows the time required by steps (1) and (2) for each of those networks. As shown in the table, those times are negligible (less than $50\ ms$), for all simulated network sizes.

Because Gibbs sampler is a Monte Carlo-based statistical algorithm, it terminates when the number of sampling iterations it performs produces estimated probabilities that converge according to a given error threshold. Fig. 7 shows the time requirements for the Gibbs sampler (GS) inference procedure (step (3)) in the four evaluated WSNs, considering two distinct number of iterations commonly used in the literature [Raftery and Lewis 1992], namely, $2,000$ and $10,000$. As presented in the figure, in a small WSN comprised of 20 nodes, the security metric computation for all nodes takes 0.25 and 1.2 seconds, in the two respective number of Gibbs Sampler iterations evaluated. On the other hand, in a reasonably large network of 100 nodes, the inference time increases to 2.8 and 13.1 seconds, respectively.

In summary, the experiments have shown that for typical WSNs containing from 20 to 100 nodes, the times required to perform the steps of S3 are fairly acceptable. On the other hand, it is also important to highlight that because of the properties of Bayesian networks, the inference times may increase exponentially as the network size grows. However, a number of iterations as low as $2,000$ can be used to carry out BN inferences in a timely fashion (for larger networks), at the cost of providing slightly less accurate security metric values (as will be shown in the next section).

4.5. Accuracy of the Estimated Security Metric

Since Gibbs sampler is an approximation algorithm, this section evaluates how incorrect can be the security metric values estimated by S3 when compared to the actual security status of nodes. In particular, various scenarios were carried out where a worm propagates through the network infecting several nodes. During the experiments, the number of times each node has been compromised by the worm is counted. Then, the fraction of times each node is compromised is compared to the probability value provided by S3 for each node. The results of the comparisons are shown in terms of the absolute error of the estimated value, i.e., the absolute difference between the actual infection probability and

Number of Nodes	APG generation (ms)	BN generation (ms)
20	0.007	2
30	0.015	4
50	0.025	8
100	0.068	45

Figure 6. Time requirements of S3 steps.

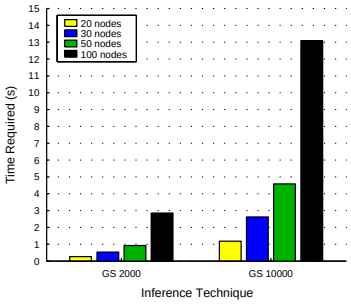


Figure 7. Time requirement of S3 inference technique.

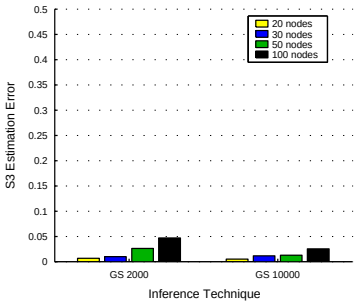


Figure 8. Error of S3 security metric estimation.

the infection probability estimated by S3. Fig. 8 presents those results for two distinct amounts of Gibbs sampler iterations and four different sized networks.

As shown in the figure, the average error in the estimated value is slightly smaller when the number of iterations is larger. Even in the worst case, the error value is fairly small, i.e., approximately 0.05 (for GS $2,000$ and 100 nodes). This error is even smaller (i.e., around 0.02) when $10,000$ Gibbs sampler iterations are performed.

In summary, the error values can be considered small for all network sizes and number of Gibbs sampler iterations. Hence, in the case of very large networks, it may be worth to decrease the number of Gibbs sampler iterations so as to improve performance (computation overhead), while still obtaining accurate S3 estimates.

5. Related Work

Most of the existing works on security quantification are focused on traditional networks. Those proposals are usually based attack graphs, which measure security based on the interdependency of system vulnerabilities (*e.g.,* the Weakest Adversary [Pamula et al. 2006] and the Attack Resistance [Wang et al. 2007] metrics). However, none of those proposals is suitable to quantify security in sensor networks due to the specific characteristics, vulnerabilities, and attack types of WSNs [Walters et al. 2007].

So far, only a few works have been proposed specifically for WSNs [Anand et al. 2005, Ramos and Filho 2015]. Anand et al. [Anand et al. 2005] propose a model that probabilistically quantify the resilience of WSN protocols against eavesdropping attacks. Their model is based on information such as sensor data distribution and topologies. However, their model is designed to evaluate security statically, rather than in an online manner.

An online security quantification scheme for WSNs has been recently proposed by Ramos et al. [Ramos and Filho 2015]. This scheme is based on three security metrics that respectively measure the resilience of the three main security mechanisms deployed in WSNs (*i.e.,* cryptography, key management, and IDS). Although this scheme addresses several attack types and considers IDS alerts, it treats attacks as independent events and, consequently, disregards worm attacks.

Finally, there exist some works [Haghighi et al. 2016, De et al. 2009] that use epidemic theory to model worm propagation in WSNs. Those proposals provide useful information that enable to understand worm attack behavior as well as to develop defensive

strategies. However, such works are not suitable for an online security evaluation since they are usually based on abstract input parameters (which are very difficult to obtain) and focus on the static analysis of the WSNs, rather than on the operational analysis.

Unlike the previous existing works, the proposed S3 model is a practical approach, which has been implemented and evaluated in a real WSN operating system. Furthermore, by using the Bayesian networks formalism, S3 is able to capture the dependency among the attacks that occur in different nodes in the WSN and evaluate node security level in an online manner. It should be noted that S3 has been mostly inspired by the security evaluation framework proposed in [Zonouz et al. 2015] for energy delivery systems.

6. Conclusion

This paper presented the Sensor Security Status (S3), a security metric model for estimating the security level of sensor nodes. Since WSNs are vulnerable to worm attacks, which can take advantage of the network communication pattern to spread throughout the network, S3 combines IDS alerts with the communication behavior of sensor nodes to estimate the probability that a node has been compromised given that other malicious nodes are present in the network. The presented simulation results show that S3 accurately represents node security level while keeping energy and performance overhead low.

Since IDSs may have false positives, an interesting future work would be to extend S3 to deal with IDS inaccuracies. This could be done by integrating S3 with the IDS effectiveness metric proposed in [Ramos et al. 2017]. Furthermore, considering that the WSN topology may change over time (e.g., new nodes can be added), it would also be interesting to develop an approach to periodically rebuild the APG to reflect the changes in the network communication graph while still keeping node's energy consumption low. Another future work is to consider other worm propagation strategies.

Acknowledgments

This work has been partially supported by Coordination for the Improvement of Higher Education Personnel (CAPES), Instituto de Telecomunicações, Next Generation Networks and Applications Group (NetGNA), Covilhã Delegation, by National Funding from the FCT - Fundação para a Ciência e a Tecnologia through the UID/EEA/500008/2013 Project, and by Finep, with resources from Funttel, grant no. 01.14.0231.00, under the Radiocommunication Reference Center (Centro de Referência em Radiocomunicações CRR) project of the National Institute of Telecommunications (Instituto Nacional de TelecomunicaçõesInatel), Brazil.

References

Anand, M., Ives, Z., and Lee, I. (2005). Quantifying eavesdropping vulnerability in sensor networks. In *ACM DMSN Wksp*, pages 3–9.

Casella, G. and George, E. I. (1992). Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174.

De, P., Liu, Y., and Das, S. K. (2009). Deployment-aware modeling of node compromise spread in wireless sensor networks using epidemic theory. *ACM Trans. Sen. Netw.*, 5(3):23:1–23:33.

Dunkels, A., Eriksson, J., Finne, N., and Tsiftes, N. (2011). Powertrace: Network-level Power Profiling for Low-power Wireless Networks. Technical Report 4112.

Dunkels, A., Gronvall, B., and Voigt, T. (2004). Contiki - a lightweight and flexible operating system for tiny networked sensors. In *IEEE LCN Conf.*, pages 455–462.

Francillon, A. and Castelluccia, C. (2008). Code injection attacks on harvard-architecture devices. In *ACM CCS Conf.*, pages 15–26.

Giannetsos, T., Dimitriou, T., and Prasad, N. R. (2009). Self-propagating worms in wireless sensor networks. In *ACM Co-Next Student Wksp*, pages 31–32.

Haghighi, M. S., Wen, S., Xiang, Y., Quinn, B., and Zhou, W. (2016). On the race of worms and patches: Modeling the spread of information in wireless sensor networks. *IEEE Transactions on Information Forensics and Security*, 11(12):2854–2865.

Ho, J.-w. (2015). Distributed Software-Attestation Defense against Sensor Worm Propagation. *Journal of Sensors*, 2015:1–6.

Khayam, S. A. and Radha, H. (2005). A topologically-aware worm propagation model for wireless sensor networks. In *IEEE ICDCS Wksps*, pages 210–216.

King, D. (2002). Dlibc++. Available: `http://dlib.net`. Accessed: 2016-12-08.

Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., and Voigt, T. (2006). Cross-level sensor network simulation with cooja. In *IEEE LCN Conf.*, pages 641–648.

Pamula, J., Jajodia, S., Ammann, P., and Swarup, V. (2006). A weakest-adversary security metric for network configuration security analysis. In *QoP Wksp*, pages 31–38.

Polastre, J., Szewczyk, R., and Culler, D. (2005). Telos: enabling ultra-low power wireless research. In *IPSN International Symposium*, pages 364–369.

Raftery, A. E. and Lewis, S. (1992). How many iterations in the Gibbs sampler. In *In Bayesian Statistics 4*, volume 4, pages 763–773.

Ramos, A. and Filho, R. (2015). Sensor Data Security Level Estimation Scheme for Wireless Sensor Networks. *Sensors*, 15(1):2104–2136.

Ramos, A., Lazar, M., Holanda Filho, R., and Rodrigues, J. J. P. C. (2017). A Security Metric for the Evaluation of Collaborative Intrusion Detection Systems in Wireless Sensor Networks. In *IEEE International Conference on Communications (Accepted for publication)*.

Raza, S., Wallgren, L., and Voigt, T. (2013). SVELTE: Real-time intrusion detection in the Internet of Things. *Ad Hoc Networks*, 11(8):2661–2674.

Walters, J., Liang, Z., Shi, W., and Chaudhary, V. (2007). *Wireless Sensor Network Security: A Survey*, page 367. CRC Press: Boca Raton, FL, USA.

Wang, L., Singhal, A., and Jajodia, S. (2007). Measuring the overall security of network configurations using attack graphs. In *DBSec Conf.*, pages 98–112. Springer.

Winter, T., Thubert, P., Brandt, A., Hui, J. W., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., and Alexander, R. (2012). RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. Technical Report 6550.

Zonouz, S. A., Berthier, R., Khurana, H., Sanders, W. H., and Yardley, T. (2015). Seclius: An Information Flow-Based, Consequence-Centric Security Metric. *IEEE Transactions on Parallel and Distributed Systems*, 26(2):562–573.

Trilha Principal do SBRC 2017
Sessão Técnica 15
Redes Definidas por Software II

Mobilidade transparente em redes mesh sem fio definidas por software

Italo Brito[1], **Gustavo B. Figueiredo**[1]

[1]Departamento de Ciência da Computação – Universidade Federal da Bahia (UFBA)
Av. Adhemar de Barros, s/n – Campus Ondina – Salvador, BA – Brasil

{italo,gustavo}@dcc.ufba.br

Abstract. *The ever increasing mobile data traffic has made the support for seamless mobility an imperative requirement for network providers. To deliver enhanced mobility experience, the handoff process must avoid interruption of connections while preserving QoS and QoE at acceptable levels. However, traditional mobility strategies based on the 802.11 standard impose high latency and QoS degradation, deteriorating the application performance. In this paper we present a new strategy to provide seamless mobility to unmodified clients in wireless mesh networks that considers network controlled handoff and passive measurements. Experimental results show the effectiveness of the proposed strategy in providing seamless handoff, improving QoE up to 70% while keeping key QoS metrics below real-time applications constraints.*

Resumo. *O uso de dispositivos móveis tem crescido de tal maneira que o suporte a mobilidade transparente é requisito crucial para os provedores de rede. De modo a proporcionar uma boa experiência de mobilidade aos usuários, o processo de handoff deve preservar a conectividade e bons níveis de QoS e QoE. Por outro lado, as estratégias de gerenciamento de mobilidade baseadas no padrão 802.11 incorrem em latência e degradação da QoS na rede, impactando negativamente o desempenho das aplicações. Este trabalho apresenta uma estratégia de handoff definido por software e baseado em monitoramento passivo da qualidade dos enlaces, onde o processo de handoff é inteiramente controlado pela rede. Resultados experimentais em um testebed mostram uma melhora de até 70% nas métricas de QoE durante a mobilidade de clientes, ao passo que métricas chave de QoS mantiveram-se abaixo de limites de restrição para aplicações de tempo real.*

1. Introdução

O número de dispositivos portáteis (e.g. *notebooks*, *tablets*, *smartphones*) conectados às redes sem fio tem crescido sobremaneira nos últimos anos [Lee et al. 2016]. Esse crescimento, associado à necessidade dos usuários estarem sempre conectados, torna clara a demanda por mobilidade eficiente entre os clientes móveis (MC, do inglês *Mobile Client*) e os pontos de acesso (AP, do inglês *Access Point*) sem fio. O processo de migração da associação entre MCs e APs, conhecido como *handoff* ou *handover*, idealmente, deve ser transparente para os nós móveis, dispensando configurações ou protocolos adicionais e mantendo a conectividade sem interrupções. Deve-se, portanto, preservar a Qualidade de Serviço (QoS) das conexões durante o *handoff*, evitando atraso excessivo, variação

de atraso e perda de pacotes. Além disso, deve-se também manter a Qualidade de Experiência (QoE) dos usuários em níveis aceitáveis, sem deteriorar a qualidade de transmissão na percepção dos usuários. A observância desses requisitos é particularmente importante para suporte a aplicações de tempo real e de multimídia interativa.

As técnicas tradicionais de suporte à mobilidade em redes sem fio, incluindo aquelas disponíveis para dispositivos que adotam o padrão 802.11 (*Wi-Fi*), baseiam-se no processo de *handoff* monitorado e executado pelo cliente. No entanto, trabalhos anteriores [Amir et al. 2010, Schulz-Zander et al. 2014] mostram que essa estratégia implica em atrasos de grandes proporções na rede. Ademais, como o processo de *handoff* é controlado pelo cliente, não existem garantias de que ele seja iniciado no momento correto. Isto pode impactar negativamente no desempenho do cliente, bem como, na rede como um todo, levando a congestionamento no AP, retransmissões excessivas e perda de pacotes. Outros mecanismos [Croitoru et al. 2015, Dely et al. 2011] requerem modificações ou inserção de protocolos adicionais no cliente, tornando-se um obstáculo para sua ampla adoção [Schulz-Zander et al. 2014].

Este trabalho apresenta uma abordagem para mobilidade transparente em redes *mesh* sem fio, por meio do gerenciamento de *handoff* baseado no paradigma de Redes Definidas por Software (SDN, do inglês *Software-Defined Networking*) e monitoramento passivo da qualidade dos enlaces Wi-Fi. O processo de *handoff* é inteiramente controlado pela rede, o que viabiliza mobilidade otimizada a partir de melhores decisões do ponto de vista global e mantém os dispositivos clientes livres de configurações adicionais. Foi realizada uma avaliação de desempenho detalhada da proposta, em comparação com uma implantação baseada no protocolo B.A.T.M.A.N. [Johnson et al. 2008] e *handoff* controlado pelo cliente, utilizando o padrão 802.11. Levou-se em consideração métricas de QoS e QoE para avaliar o impacto do *handoff* na transmissão de dados. Resultados mostram que a proposta de gerenciamento de *handoff* melhorou até 70% as métricas de QoE durante a mobilidade de clientes, ao passo que manteve métricas chave de QoS abaixo de limites de restrição para aplicações de tempo real.

O restante do trabalho está divido da seguinte forma. A Seção 2 traz um resumo de trabalhos correlatos. A Seção 3 apresenta a estratégia de *handoff* proposta neste trabalho. Em seguida, na Seção 4, experimentos comparativos são apresentados e os resultados são discutidos. Por fim, na Seção 5 apresente-se as conclusões e trabalhos futuros.

2. Trabalhos relacionados

Dely et al. [Dely et al. 2011] foi um dos primeiros a integrar SDN em redes *mesh*, propondo uma arquitetura para controle e monitoramento baseado em encaminhamento por fluxos. Como estudo de caso, foi implementada uma função de gerenciamento de mobilidade onde o controlador monitora continuamente a qualidade dos enlaces e a capacidade dos APs, disparando uma ação de *handoff* no cliente, através do protocolo IEEE 802.21, quando detecta a mobilidade. Além do suporte a 802.21, os clientes devem ser modificados para instalação do agente de monitoramento ativo da rede sem fio. Diferente do trabalho supracitado, a proposta aqui apresentada é independente de alterações no cliente e realiza o monitoramento passivo da rede.

Em termos de padronizações, o IEEE tem abordado o *handoff* de redes IP por meio de diferentes normas e complementos. O complemento 802.11k, por exemplo, re-

duz o tempo do *handoff* pois permite que o cliente determine mais rapidamente a qual AP ele deve se associar, através de informações sobre a vizinhaça fornecida pelo AP atual. Já o 802.11r otimiza o tempo de autenticação pelo uso de *cache* das chaves de criptografia nos APs. O complemento 802.11v permite que os APs compartilhem com o cliente informações de sobrecarga da rede, a fim de melhorar a escolha do cliente pelo novo AP. Já o padrão 802.21 visa prover mobilidade entre diferentes tipos de rede, também conhecido como *handoff independente de mídia*. Todas essas abordagens requerem alterações no padrão 802.11 e, consequentemente, nos APs e nos clientes. Neste trabalho, nenhuma modificação no padrão 802.11 é necessária.

Avelar [Avelar 2013] propõe uma arquitetura baseada em SDN para gerenciamento de mobilidade em redes IP inspirada no protocolo PMIPv6 (do inglês *Proxy Mobile IPv6*), chamada PMIPFlow. Apresenta-se também um mecanismo de antecipação de *handoff* baseado na lógica *fuzzy* (PMIPFlow-Fuzzy) para proporcionar mobilidade transparente. A antecipação do *handoff* é possível para clientes que executam um agente do PMIPFlow-Fuzzy, que utiliza três variáveis (RTT, Vazão e RSSI) na criação das regras de inferência do sistema *fuzzy*. Demonstrou-se que a proposta de antecipação do *handoff* permite mobilidade transparente e com baixo impacto na qualidade de experiência de aplicações como vídeo *streaming*. Todavia, a necessidade de instalação de um agente PMIPFlow-Fuzzy pode se tornar uma barreira para amplo benefício dos nós clientes. Além disso, o monitoramento ativo pode gerar sobrecarga adicional na rede.

Schulz-Zander et al. [Schulz-Zander et al. 2014] propõem uma solução para orquestração de redes *Wi-Fi* com SDN, chamada Odin. Entre as contribuições do trabalho, destaca-se a proposta de pontos de acesso virtuais leves (LVAPs, do inglês *Light Virtual Access Points*), onde cada cliente associa-se com um AP virtual específico, através de um BSSID único. Para manter a associação, os LVAPs enviam periodicamente *beacons* específicos do cliente, inclusive ao longo de uma trajetória de movimentação. Assim, migrando o LVAP de cada cliente entre os APs físicos, a infraestrutura pode então controlar o *handoff* do cliente, sem modificá-lo e sem técnicas de sinalização de canais adicionais. Uma abordagem semelhante, porém simplificada, foi utilizada neste trabalho.

3. Estratégia de *handoff* transparente

Neste trabalho é proposta uma nova estratégia de gerenciamento de *handoff* transparente e controlado pela rede. O gerenciamento do *handoff* consiste em manter a conectividade de clientes móveis quando eles estão nos limites de cobertura de sinal de dois ou mais APs. O *handoff* ocorre quando a potência do sinal (RSSI), ou qualquer outra métrica de qualidade do enlace *Wi-Fi*, degrada até certos limiares mínimos. O processo do *handoff* é composto pelas etapas: i) monitoramento, onde realiza-se medições sobre a potência de sinal, distância, qualidade, taxa de erros, etc; ii) decisão, onde compara-se os valores medidos com os limiares configurados, podendo levar em consideração margens de confiança para evitar efeitos de *ping-ponging* no *handoff* [Chandra Paul 2014]; e iii) execução, consistindo da sinalização do *handoff*, alocação de recursos nos rádios, re-estabelecimento de rotas na rede.

A estratégia ora proposta traz contribuições principalmente para o monitoramento e decisão de *handoff*, com uma combinação inédita de métricas obtidas de forma passiva na rede e um algoritmo de processamento das métricas para escolha do melhor AP

para o nó móvel. A proposta foi implementada no OpenWiMesh [Brito et al. 2014], um *framework* para orquestração de redes *mesh* sem fio com SDN, permitindo a programabilidade da rede *mesh* com base em um grafo que modela as características da rede.

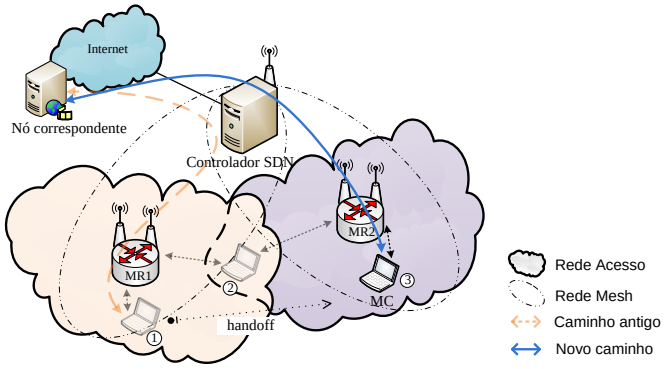


Figura 1. Estratégia de mobilidade e *handoff* transparente no OpenWiMesh

A estratégia de mobilidade transparente aqui apresentada é ilustrada na Figura 1. Nesta figura, é possível notar a interação entre MC, roteadores *mesh* (MR) e Controlador SDN (CTL) nas diferentes fases do processo de *handoff*: na fase 1, o MC está associado com o MR1 que é o único AP alcançavel naquela zona *Wi-Fi*; ao se mover em direção à direita (fase 2), o MC passa a ser alcançável também pelo MR2 e tem sua qualidade de enlace monitorada por aquele roteador *mesh*; o MC continua a mover-se até que, na fase 3, a rede detecta que MR2 possui melhor qualidade para atendê-lo e executa o *handoff*. É também na fase 3 que as rotas do MC são reconfiguradas, como no exemplo ilustrado, onde a rota entre o MC e um nó correspondente (e.g. servidor de vídeo na Internet) que antes passava por MR1 e CTL agora passa por MR2 e CTL.

Todo o processo do *handoff*, mecanismos de monitoramento, detecção e execução serão detalhados nas subseções seguintes.

3.1. Métricas de qualidade de enlaces *Wi-Fi*

Alguns *drivers* de interfaces *Wi-Fi* do Kernel Linux suportam a API *nl80211*[1], que permite obter informações específicas do meio sem fio, a partir de frames capturados pelo dispositivo *Wi-Fi*. Algumas dessas informações são:

- **Received Signal Strength Indicator (RSSI).** RSSI é uma medição da potência de sinal recebido pelo rádio. Os pacotes de controle da rede *Wi-Fi* (*beacons*) por padrão incluem essa informação e o *framework cfg80211* do Kernel Linux torna essa informação disponível através da API *nl80211*, disponibilizando o RSSI de cada associação (tanto no modo *ad-hoc* quanto infraestruturado).
- **802.11 Retry Counter (RTY).** Cada pacote 802.11 transmitido em unicast necessita de uma confirmação (ACK) do receptor. Se o pacote ou o ACK forem perdidos na transmissão, o emissor o retransmite e incrementa o contador de retransmissões. Através da API *nl80211* é possível obter estatísticas sobre a quantidade total de pacotes transmistidos, retransmissões, quantidade de pacotes que falharam (esgotou-se o limite de retransmissão), etc., para cada nó vizinho.

[1]https://wireless.wiki.kernel.org/en/developers/documentation/nl80211

Diferente de trabalhos anteriores, que consideram estas métricas de forma isolada [Amir et al. 2010, Chandra Paul 2014, Tsukamoto et al. 2007], aqui propõe-se seu uso combinado para calcular a qualidade do enlace *Wi-Fi*. Destarte, o RSSI contabiliza a potência de sinal do *uplink* do cliente e a taxa de retransmissão permite a inferência da qualidade no enlace de *downlink*. Ambas as métricas são monitoradas de forma passiva no AP, sem sobrecarga adicional. Além disso, as taxas de transmissão e recepção de pacotes, TX_{pps} e RX_{pps} (expressas em *pacotes por segundo*), respectivamente, são aplicadas para balancear essas métricas, atribuindo pesos para o RSSI e RTY de acordo com o volume e direção do tráfego. Assim, ajusta-se a importância de cada métrica de forma dinâmica e equilibrada. Os pesos de transmissão ($Weight_{tx}$) e recepção ($Weight_{rx}$) são definidos como:

$$Weight_{tx} = \frac{TX_{pps}}{TX_{pps} + RX_{pps}} \quad (1) \qquad Weight_{rx} = \frac{RX_{pps}}{TX_{pps} + RX_{pps}} \quad (2)$$

Para que as métricas RSSI e RTY possam ser combinadas, elas são, antes de tudo, normalizadas. A Equação 3 apresenta a normalização do valor do RSSI (medido em dBm) para porcentagem da qualidade do sinal ($Signal_Q$). Já a normalização da taxa de retransmissão (FRR) leva em consideração a taxa de pacotes retransmitidos (RTY, medido em pps) em comparação com o total de transmitidos, conforme Equação 4.

$$Signal_Q = \frac{2 * (dBm + 100)}{100}, dBm \in [-100, -50] \quad (3)$$

$$FRR = \frac{RTY_{pps}}{TX_{pps} + RTY_{pps}} \quad (4)$$

A qualidade do enlace *Wi-Fi* é expressa na Equação 5. Nessa equação, além de considerar o RSSI e RTY, balanceados pelos valores de $Weight_{rx}$ e $Weight_{tx}$, considera-se também o tempo de inatividade das estatísticas informadas pelo *nl80211*. Quando um nó deixa de receber pacotes de controle (*beacons*) de determinado vizinho (por degradação do sinal ou movimentação do nó), o tempo de inatividade passa a incrementar diminuindo a importância das estatísticas.

$$Link_Q = \frac{Weight_{tx} * (1.0 - FRR) + Weight_{rx} * Signal_Q}{inactiveTime}, inactiveTime > 0 \quad (5)$$

A métrica acima produz uma visão instantânea da qualidade do sinal *Wi-Fi*. Não obstante, devido às oscilações do meio sem fio, esses valores podem variar e causar o efeito *ping-ponging* [Chandra Paul 2014]. Para evitar essas oscilações, a programação dessas funções pode fazer uso da média do RSSI no controlador, utilizar a média de RSSI disponibilizada pela API *nl80211* ou ainda aplicar uma média móvel ao valor de $Link_Q$. Neste trabalho utilizou-se uma média móvel de tamanho σ, arbitrado em 5.

3.2. Algoritmo de decisão de *handoff*

A métrica de qualidade do enlace previamente apresentada é calculada a cada atualização do grafo da rede, cujo intervalo padrão é um segundo. Após o cálculo da qualidade dos enlaces ($Link_Q$) de todos os clientes associados a um nó, uma rotina de checagem da necessidade do *handoff* é chamada. Essa rotina verifica se o $Link_Q$ está abaixo de determinado liminar T (*threshold*) e se existe outro vizinho com $Link_Q$ melhor que a associação atual e com diferença H (histerese). Essa abordagem é semelhante à apresentada em [Chandra Paul 2014], chamada de *Relative Signal Strength with Threshold and Hysteresis*, porém aqui expandida considerando outras métricas (Equação 5).

O pseudo-código da Listagem 1 apresenta a lógica de verificação da qualidade do enlace e o algoritmo de decisão de *handoff* supracitado.

```
pooler_check_assoc_list(node):
    assoc_list = graphClient.get_assoc_list(node)
    FOR neigh IN assoc_list:
        IF link_quality(node, neigh) < T:
            target_ap = check_handoff(node, neigh)
            IF NOT target_ap.empty():
                execute_handoff(neigh, target_ap)

check_handoff(cur_ap, client):
    eligible_cand = []
    cur_linkq = link_quality(cur_ap, client)
    FOR cand IN client.neighbors():
        IF link_quality(cand, client) > cur_linkq + H:
            capacity = calc_capacity(cand)
            eligible_cand.append([cand, capacity])
    sorted_eligible_cand = sort(eligible_cand)
    RETURN sorted_eligible_cand.first()
```

Listagem 1. *Pseudo-código das rotinas de detecção e decisão de handoff*

A escolha de qual AP melhor atenderá ao cliente em *handoff* deve levar em consideração a capacidade dos APs elegíveis para suportar o novo cliente. Para o cálculo da capacidade disponível em um AP, pode-se levar em consideração diferentes métricas como vazão (máxima teórica, mínima, vazão derivada), CPU ou memória disponível no AP, quantidade de clientes associados, etc. Neste trabalho optou-se por utilizar a média de banda residual entre todas as interfaces do AP *mesh*, conforme apresenta a Equação 6. Embora a utilização de outras métricas possa melhorar a estimativa de capacidade de um nó, em testes de laboratório realizados, foi possível constatar que o uso da banda residual possui fidelidade com a capacidade do nó, enquanto que seu cálculo é simplificado.

$$calc\_capacity = \frac{1}{n} \sum_{iface=1}^{n} (1 - \frac{cur\_throughput_{iface}}{max\_speed_{iface}}) \quad (6)$$

3.3. Execução do *handoff*

Ao detectar a necessidade de *handoff* e escolher o novo AP de um cliente móvel, a estratégia proposta inicia as rotinas de reconfiguração de rotas para as aplicações do cliente. Para isso, são mantidas informaçõs sobre as aplicações e rotas ativas na rede, indexando-as nos arcos do grafo. Dessa maneira, a execução do *handoff* consiste em processar a lista de aplicações em cada arco entre o cliente e o AP atual, buscando novo caminho no grafo que contemple o novo AP e reconfigurá-las em cada MR do caminho. A reconfiguração das rotas ocorre através de mensagens Openflow do tipo *flow-mod*.

Em termos da associação *Wi-Fi*, como foi utilizado um único BSSID nos APs, não é necessária sinalização adicional ou re-associação no cliente. Para configurar os nós com essa característisca basta manter-se o mesmo SSID, BSSID, canal e, caso utilize criptografia, chave da sessão. Neste trabalho foi utilizado o software AP *hostapd*[2] para criação da rede *Wi-Fi* infraestruturada dos clientes.

[2]O *hostapd* é um software que transforma uma interface de rede *Wi-Fi* convencional em um AP.

A reconfiguração de rotas ocorre de forma a primeiro criar as novas rotas, para somente então remover as rotas antigas. A ordem de execução do *handoff* é importante para diminuir o impacto no tráfego das aplicações, deixando-o assim mais transparente ao cliente. Na literatura essa metodologia é conhecida como *make before break* (criar e depois desfazer) ou *soft-handoff* [Chandra Paul 2014], sendo utilizada nas redes de celulares CDMA e W-CDMA mas não nas redes 802.11. Com a metologia utilizada é possível importar esse comportamento para as redes 802.11. Ademais, a fase de remoção das rotas é postergada por alguns segundos a fim de manter o caminho antigo ativo e assim aumentar a probabilidade de recepção de pacotes no cliente. Essa abordagem pode gerar duplicação de pacotes na rede, porém seu custo benefício a torna interessante e viável, como será visto mais adiante.

4. Avaliação da proposta

Para avaliar a estratégia proposta, uma série de experimentos e medições foram conduzidos, através de um *testbed* 802.11/OpenFlow. Neste *testbed*, um conjunto de MR se comunicam através de uma rede *mesh* orquestrada pelo OpenWiMesh e possuem uma interface *Wi-Fi* adicional para acesso dos clientes (baseada na metodologia LVAP). A fim de prover uma linha base de comparação, foi utilizada uma segunda estratégia de gerenciamento de mobilidade: APs divulgando o mesmo SSID com diferentes BSSID's; roteamento entre os MRs através do protocolo B.A.T.M.A.N. (rede *mesh*). Nesse cenário, nomeado BATMAN-MCHO (*B.A.T.M.A.N. with Mobile Controlled Handoff*), os clientes móveis controlam o *handoff*. Por outro lado, no mecanismo de *handoff* proposto neste trabalho, chamado OpenWiMesh-Mob, a mobilidade é controlada pela rede e transparente para o cliente.

O *testbed* foi construído da seguinte maneira. Os MRs foram implantados a partir de máquinas com CPU Intel Core i5, 8GB de memória, S.O. Debian 8.3, interface *Wi-Fi* Atheros AR9485 (usada na rede *mesh*) e interface *Wi-Fi* TP-Link WN721N (usada no *hostapd*). O CTL executa em um notebook com hardware similar aos MRs, assumindo também o papel de Nó Correspondente (CN). O Cliente Móvel (MC) comunica-se com o CN durante os testes, utilizando o serviço de transferência de vídeo e serviço medições *iperf*. O MC foi baseado em três das plataformas mais utilizadas atualmente como sistema operacional em computadores pessoais: Linux, Windows e MAC OS X. Dessa forma, é possível validar o comportamento das estratégias com relação ao gerenciamento de *handoff* em diferentes plataformas.

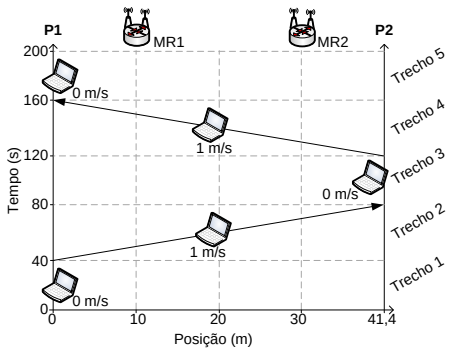
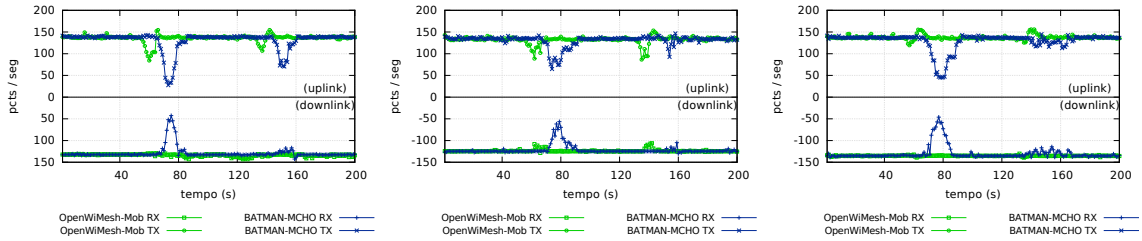


Figura 2. Diagrama de mobilidade no teste de *handoff*

O cenário utilizado é similar ao apresentado na Figura 1 e o diagrama de mobilidade é ilustrado na Figura 2. A metodologia de avaliação consistiu em testes com duração de 200 segundos, divididos da seguinte maneira: o MC começa no ponto P1 e permanece naquele ponto por 40 segundos; ele, então, move-se em direção ao ponto P2 com velocidade aproximada de 1 m/s e duração de 40 segundos; permanece no ponto P2 por mais 40 segundos; retorna a P1 com velocidade 1m/s e duração 40s; e, finalmente, permanece em P1 nos 40 segundos finais. Cada experimento foi repetido 10 vezes por plataforma.

Para medir a QoS fez-se uso de geração de tráfego baseado em taxa de bits constante, com intervalo de envio de pacotes de 8 ms e tamanho do pacote de 200 bytes, através da ferramenta *iperf*. A partir dos experimentos, foi possível aferir as seguintes métricas: taxa de comutação de pacotes, atraso, variação de atraso (*jitter*), potência de sinal *Wi-Fi* recebido e taxa de retransmissão de pacotes.

O gráfico da Figura 3 apresenta os resultados para taxa de comutação de pacotes. Variações na linha que representa a taxa de comutação podem ser entendidas como perda de pacotes e quanto mais próximas a zero pior. Pela análise dos gráficos é possível verificar que a abordagem de *handoff* BATMAN-MCHO possui impacto significativo na qualidade de serviço da rede, levando a grande perda de pacotes nos momentos de *handoff*, independente da plataforma em questão. Essa perda se dá principalmente pelo fato de que a decisão de *handoff* é tomada pelo cliente, que não possui conhecimento das condições da rede e toma ações de maneira tardia, com pequena variação entre sistemas operacionais. Por outro lado, o *handoff* controlado pelo OpenWiMesh-Mob fornece melhores condições de rede para o cliente, com pequenas oscilações no tráfego, porém sem causar impactos perceptíveis ao usuário na conexão. No Mac OS X, por exemplo, a taxa de comutação de pacotes no *downlink* do OpenWiMesh-Mob apresentou média de 135.30 pps, com desvio padrão de 0.93, enquanto que o BATMAN-MCHO foi de 129.96 ± 15.54 pps. Em particular, no Trecho 2, quando ocorre o primeiro *handoff*, o BATMAN-MCHO apresenta uma degradação de 14.03% em relação ao OpenWiMesh-Mob.



(a) Linux (b) Windows (c) Mac OS X

Figura 3. Taxa de comutação de pacote entre MC e CN

Nas Figuras 4a e 4b é possível verificar o atraso bidirecional e variação de atraso na rede, utilizando o mesmo modelo de tráfego anteriormente citado. Algumas aplicações são bastante sensíveis ao atraso e variação do atraso, tolerando atraso máximo de 100ms e variação de 20ms a fim de garantir alta qualidade e interatividade [ITU-T 2003, Amir et al. 2010]. Pela Figura 4a e 4b é possível observar que o OpenWiMesh-Mob esteve abaixo desses limites nas métricas de atraso e variação de atraso. Em relação ao atraso, o OpenWiMesh-Mob supera a estratégia BATMAN-MCHO

em cerca de 300% durante as etapas de *handoff*. Já com relação à variação de atraso, o OpenWiMesh-Mob teve um *jitter* de *downlink* e *uplink* na ordem de 1.76 ± 0.57 e $3.03 \pm 2.65ms$ (média e desvio padrão), com máximo de 3.77 e $14.57ms$, respectivamente. Ao passo que o BATMAN-MCHO apresentou *jitter* de *downlink* e *uplink* de 5.55 ± 11.44 e $7.47 \pm 15.65ms$, com máximo de 89.97 e $105.32ms$, respectivamente. Devido a limitação de espaço e semelhança dos dados, apenas as métricas do Mac OS X foram apresentadas.

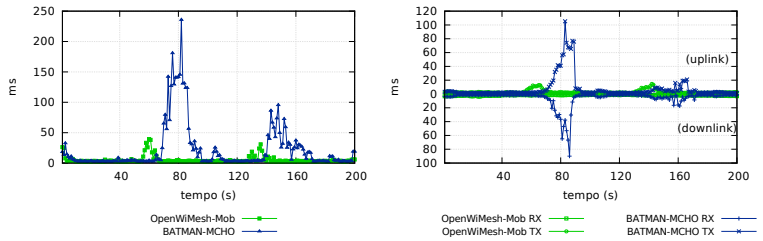


Figura 4. Avaliação de QoS: (a) atraso bidirecional; (b) *jitter*

A métrica de potência de sinal *Wi-Fi* recebido (RSSI), ilustrada através da Figura 5, possui relação indireta com a qualidade de serviço da rede, impactando na taxa de erros, atraso, jitter e probabilidade de recebimento de pacotes. As curvas do gráfico mostram que o OpenWiMesh-Mob fez boa alocação de cliente para AP, fornecendo RSSI no AP e no cliente sempre superiores a -72.1 e -67.8 dBm, sobretudo se comparado com a abordagem BATMAN-MCHO cujos valores de RSSI no AP e no cliente chegaram a -84.4 e -78.3 dBm, faixas de valores de menor qualidade do enlace.

Pela Figura 5a pode-se notar que no Trecho 2 o RSSI melhora linearmente à medida que o cliente se aproxima do MR1, depois degrada notadamente quando se distância de MR1 e acontece o *handoff* para MR2, então ele experimenta nova melhoria ao aproximar-se de MR2, e volta a cair até estabilizar no afastamento de MR2 e parada no ponto 2. Em contrapartida, a curva de RSSI do BATMAN-MCHO (Figura 5b) mostra que o sinal *Wi-Fi* se degrada à níveis muito baixos até que o cliente faça o *handoff*, que praticamente só ocorre quando ele já está no ponto 2. Comportamento similar é notado no Trecho 4 em ambas as estratégias.

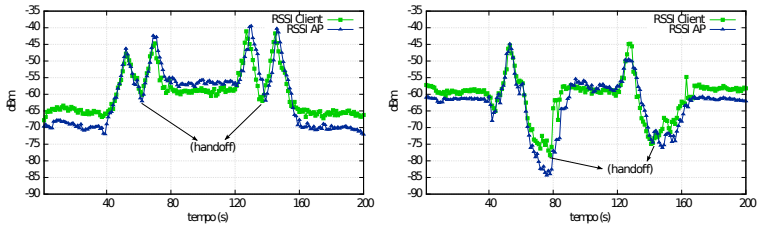


Figura 5. Potência de Sinal Recebido (RSSI): *esquerda*, (a) OpenWiMesh-Mob; *esquerda*, (b) BATMAN-MCHO

Outra métrica que indiretamente impacta na qualidade da rede é a taxa de retransmissão de pacotes da camada MAC do 802.11. Na Figura 6 é possível verificar as retransmissões de pacotes, na perspectiva do cliente e do AP, ambos com o mesmo modelo de tráfego usado até aqui. É possível notar que, independente do trecho e da existência de

handoff, há uma taxa considerável de retransmissão de pacotes, possivelmente ocasionada pelas interferências externas, uso intenso da rede no *uplink* e *downlink*, dentre outros fatores. No entanto, o OpenWiMesh-Mob consegue manter esse valor dentro de uma faixa aceitável mesmo durante a execução dos *handoff*'s nos Trechos 2 e 4. Enquanto isso, o mal gerenciamento do *handoff* é evidenciado na outra estratégia que apresenta taxa de retransmissão aproximadamente 50% maior nos trechos em que ocorre *handoff*, contabilizando média de 88.56 pps de retransmissões contra 59.53 pps do OpenWiMesh-Mob.

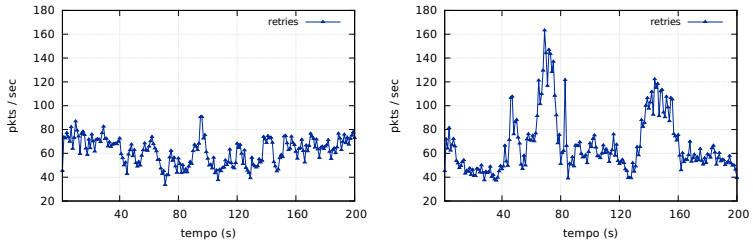


Figura 6. Taxa de retransmissão de pacotes: *esquerda*, (a) OpenWiMesh-Mob; *esquerda*, (b) BATMAN-MCHO

Para medir a QoE do usuário na utilização da rede, foi realizado um experimento com uma aplicação de *streaming* de vídeo, onde o cliente móvel assistia a um filme enquanto se movia pela rede. Para fazer o *streaming* de vídeo utilizou-se a ferramenta *live555*[3], transmitindo a animação Big Buck Bunny na resolução de 240p (320x240), formato MPEG4-TS, com duração de 200s e cadência de 15fps. Esse experimento também foi repetido 10 vezes no mesmo cenário anterior. O vídeo transmitido pela rede foi comparado com o original através de medições com PSNR (*Peak Signal to Noise Ratio*) e SSIM (*Structural Similarity Index*), métricas comumente utilizadas nesse tipo de estudo [Lambrecht and Verscheure 1996, Engelke et al. 2009]. Para comparação, foi utilizado a ferramenta MSU VQMT[4]. A partir do resultado dessa comparação é possível mapeá-la em um valor de MOS (*Mean Opnion Score*) [Lambrecht and Verscheure 1996, Engelke et al. 2009], conforme Tabela 1, e assim avaliar a QoE dos usuários.

MOS	(5) Excelente	(4) Bom	(3) Aceitável	(2) Pobre	(1) Ruim
PSNR (dB)	> 37	31 – 37	25 – 31	20 – 25	< 20
SSIM	> 0.90	0.77 – 0.89	0.61 – 0.76	0.38 – 0.60	< 0.38

Tabela 1. Mapeamento PSNR/MOS e SSIM/MOS

Os resultados da avaliação de QoE podem ser vistos na Figura 7. No gráfico do PSNR é possível visualizar claramente a queda de qualidade do vídeo após o primeiro *handoff*, embora o OpenWiMesh-Mob consiga manter uma média de qualidade de boa a excelente nos trechos seguintes. Já o BATMAN-MCHO apresenta queda de qualidade significativa apenas no final do Trecho 2 (por volta 70 segundos), quando o nível de sinal *Wi-Fi* e qualidade da rede já estão bastante degradados, resultando, portanto, em uma média de qualidade de experiência de pobre a ruim. Mesmo no trecho 3, onde o MC está parado, a qualidade do vídeo permanece em níveis menores, principalmente

[3]Website: `http://www.live555.com/`, Último acesso 20/12/2016
[4]Website: `http://goo.gl/Vio9qg`, Último acesso 20/12/2016

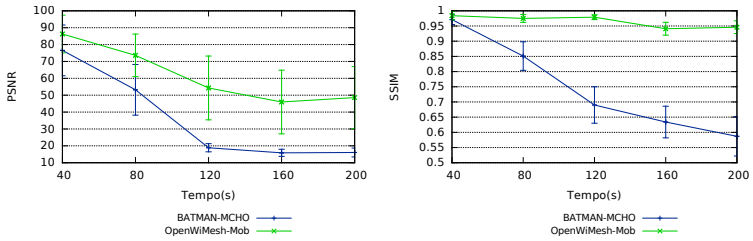


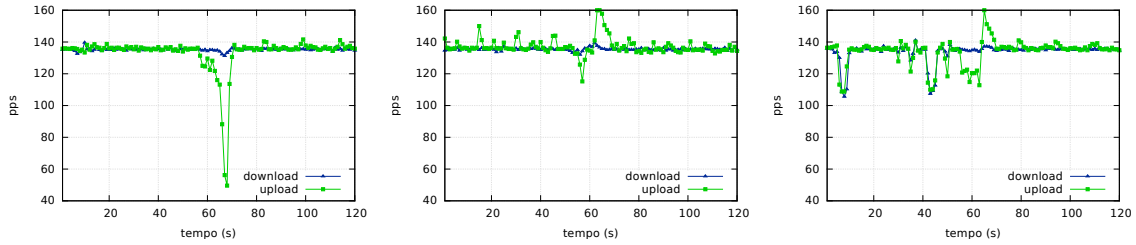
Figura 7. Qualidade de Vídeo com valores agregados: *esquerda*, (a) PSNR; *direita*, (b) SSIM

devido aos fatores de propagação de erros em frames interdependentes de vídeos MPEG [Feamster and Balakrishnan 2002]. Já a métrica SSIM revela que o OpenWiMesh-Mob consegue manter o vídeo em qualidade alta, com pequenas oscilações, e que a estratégia do BATMAN-MCHO causa queda na qualidade do vídeo variando de aceitável a pobre a partir do final do trecho 2.

É possível notar que a estratégia de *handoff* proposta apresenta desempenho médio 70% superior na métrica PSNR e 29% superior na métrica SSIM se comparado com o BATMAN-MCHO.

Realizou-se ainda um estudo comparativo acerca de diferentes parametrizações e configurações do OpenWiMesh-Mob. O objetivo dessa comparação é fornecer detalhes e identificar tendências em relação a configuração do algoritmo de *handoff* e a metodologia de teste. Devido a limitações de espaço, serão apresentados apenas a avaliação do limiar de *handoff* T, histerese de *handoff* H e padrão de mobilidade.

A fim de avaliar o impacto da escolha do valor de T, foram executados três experimentos com T assumindo os valores 0.60, 0.80 e 1.0, ambos realizados no mesmo cenário e com a mesma metodologia apresentada anteriormente, porém com duração de 120 segundos e 5 repetições. De acordo com a metodologia do experimento, o momento teoricamente ideal para o *handoff* é próximo aos 60 segundos, quando o MC está entre os dois APs e movendo em direção ao MR2.



(a) T=0.60 (b) T=0.80 (c) T=1.00

Figura 8. Taxa de comutação de pacotes com diferentes valores de T

Na Figura 8a é possível observar a taxa de comutação com $T = 0.60$, onde o *handoff* ocorre em momento ligeiramente tardio, por volta de 65 a 70 segundos e, portanto, a taxa de comutação degrada acima de 50%. Já na Figura 8c, cujo valor de T foi configurado para 1.00, percebe-se diversas oscilações ao longo do teste, em geral ocasionadas por decisões de *handoff* precipitadas. Por fim, a Figura 8b apresenta uma boa relação

de compromisso entre o limiar de *handoff* e a qualidade da rede. Nesta configuração, o algoritmo de *handoff* passa a procurar APs candidatos apenas quando o $Link_Q$ do AP atual fica abaixo de 0.8, quando as condições do enlace passam a impactar na qualidade da rede. Valores de T intermediários aos apresentados foram considerados, porém os resultados não foram representativos no cenário utilizado.

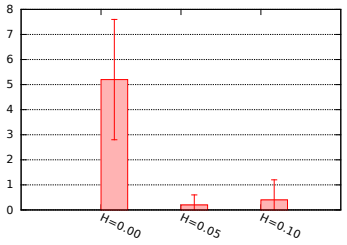
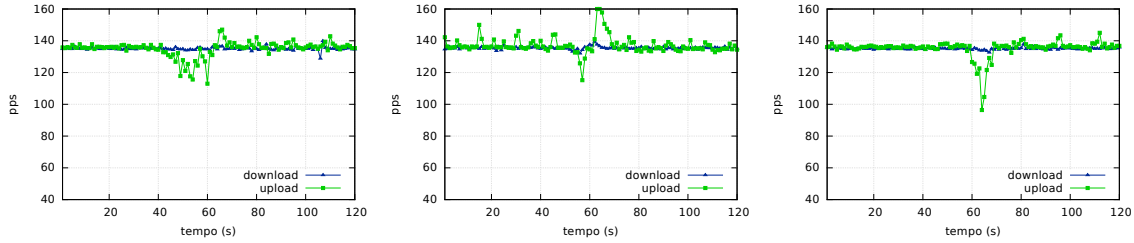


Figura 9. Quantidade de *handoffs* com diferentes valores de H

Para avaliar a relação do valor da histerese na estratégia de *handoff* proposta, foram considerados três valores de H: $H = 0.10$, $H = 0.05$ e $H = 0.00$. A metodologia de avaliação para o parâmetro H foi ligeiramente diferente das demais: o MC foi mantido estacionário entre os dois APs e mediu-se a quantidade de *handoffs* que a estratégia executou. Idealmente o cliente deve conectar a um AP e não executar nenhum ou o mínimo de *handoffs*, uma vez que ambos oferecem as mesmas condições, com variações decorrentes apenas do meio sem fio. O gráfico da Figura 9 apresenta a quantidade de *handoffs* executados pelo OpenWiMesh-Mob para cada valor de H. É possível observar que para $H = 0.00$ ocorrem muitos *handoffs* na rede, em média 5 no experimento executado, caracterizando o efeito *ping-ponging*. Ao aplicar o valor $H = 0.05$ é possível notar significativa redução na quantidade de *handoffs*, dando maior estabilidade à rede. O mesmo ocorre com $H = 0.10$. Nesse caso, o valor de $H = 0.05$ mostra-se mais vantajoso pois mantém a estabilidade da rede ao passo que oportuniza a detecção de *handoffs* válidos.



(a) Vel=0.5 m/s (b) Vel=1.0 m/s (c) Vel=2.0 m/s

Figura 10. Taxa de comutação de pacotes com diferentes velocidades do MC

Outro aspecto que merece atenção é a relação entre o padrão de mobilidade e os resultados ora apresentados. Para medir o impacto do padrão de mobilidade no processo de *handoff*, mensurou-se as métricas de QoS com MC movendo-se a $0.5m/s$, $1.0m/s$ e $2.0m/s$. Os testes foram realizados no mesmo cenário anteriormente citado. O gráfico da Figura 10 apresenta os resultados obtidos pela variação da velocidade. É possível notar ligeira diferença no momento em que o *handoff* ocorre porém sem impactos visivelmente notáveis na análise dos gráficos. Portanto, observa-se pouca relação entre o padrão de mobilidade e a estratégia de *handoff* nesse cenário.

Por fim, é crucial avaliar também o desempenho do controlador SDN, componente importante na estratégia de *handoff*. Neste trabalho foi utilizado apenas um controlador e, a fim de verificar o impacto do seu funcionamento na solução proposta, foram analisados o consumo de CPU e de memória RAM. Os testes foram realizados no mesmo cenário anteriormente citado, porém capturando dados do controlador durante 40 segundos antes do teste e 40 segundos depois. No gráfico da Figura 11a é possível observar que o consumo de CPU permanece constante e abaixo de 10% ao longo de todo o experimento, mostrando que nesse cenário um controlador foi suficiente para acomodar confortavelmente os MRs e funcionamento da rede. Já o gráfico da Figura 11b ilustra a utilização de memória no controlador, cujo consumo também manteve-se constante ao longo da execução do experimento, mesmo antes da sua inicialização e após sua conclusão.

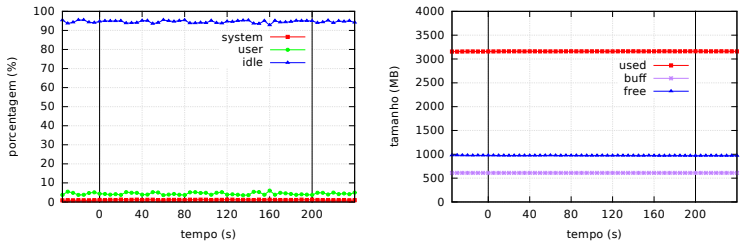


Figura 11. Desempenho do controlador SDN no OpenWiMesh-Mob: *esquerda*, (a) Consumo de CPU; *direita*, (b) Consumo de memória

5. Conclusões e Trabalhos Futuros

Esse trabalho apresentou uma proposta de métricas de monitoramento da qualidade de enlaces *Wi-Fi* e um algoritmo de decisão de *handoff* controlado pela rede, utilizando a abordagem SDN em uma rede *mesh* sem fio[5]. A avaliação comparativa aponta melhoras substanciais na qualidade da rede quando do uso da estratégia proposta. O OpenWiMesh-Mob é agnóstico a plataforma de S.O. e não requer qualquer configuração especial no cliente, viabilizando e melhorando o gerenciamento de mobilidade de dispositivos sem fio, requisito sobremaneira importante nas redes modernas.

As métricas relacionadas com a QoS da rede apontam benefícios nas estratégias aqui propostas: até 14.03% de melhoria em relação à média de taxa de comutação; redução de até 200ms no atraso birecional durante o *handoff*; variação de atraso sempre inferior a 20ms; melhor aproveitamento do sinal *Wi-Fi*; e menor sobrecarga na rede com redução na quantidade de retransmissões do 802.11. A qualidade de experiência foi avaliada em termos da qualidade de vídeo. Nas duas métricas avaliadas, PSNR e SSIM, a proposta obteve índices de qualidade média acima do aceitável em todos os momentos do teste, com avaliação global de MOS excelente, levando-se em consideração a média e desvio padrão em ambas as métricas.

Em trabalhos futuros pretende-se i) ampliar o *testbed* 802.11/OpenFlow na universidade e ii) investigar o funcionamento do *hostapd* para permitir pré-configuração dos parâmetros de associação *Wi-Fi*, evitando perdas no *uplink* durante o *handoff*.

[5]O código-fonte, scripts de configuração e dados da avaliação estão disponíveis (`http://grade.dcc.ufba.br/OpenWiMesh`, último acesso 20/12/2016) para análise e comparativos futuros, podendo ser usados ainda como base para desenvolvimento de novos algoritmos e estudos comparativos.

Referências

Amir, Y., Danilov, C., Musuăloiu-Elefteri, R., and Rivera, N. (2010). The SMesh Wireless Mesh Network. *ACM Transactions on Computer Systems (TOCS)*, 28(3):6:1–6:49.

Avelar, E. A. M. (2013). Pmipflow: Uma proposta para gerenciamento de mobilidade em redes definidas por software. Master's thesis, Universidade Federal de Pernambuco.

Brito, I., Gramacho, S., Ferreira, I., Nazaré, M., Sampaio, L., and Figueiredo, G. (2014). Openwimesh: um framework para redes mesh sem fio definidas por software. In *32th Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 413–426.

Chandra Paul, L. (2014). Handoff/handover mechanism for mobility improvement in wireless communication. *Global Journal of Researches In Engineering*, 13(16).

Croitoru, A., Niculescu, D., and Raiciu, C. (2015). Towards wifi mobility without fast handover. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 219–234.

Dely, P., Kassler, A., and Bayer, N. (2011). Openflow for wireless mesh networks. In *20th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–6. IEEE.

Engelke, U., Kusuma, M., Zepernick, H.-J., and Caldera, M. (2009). Reduced-reference metric design for objective perceptual quality assessment in wireless imaging. *Signal Processing: Image Communication*, 24(7):525–547.

Feamster, N. and Balakrishnan, H. (2002). Packet loss recovery for streaming video. In *12th International Packet Video Workshop*, pages 9–16. PA: Pittsburgh.

ITU-T (2003). One-way transmission time. *Recommendation G*, 114. Último acesso em 20 de Abril de 2016.

Johnson, D., Ntlatlapa, N., and Aichele, C. (2008). Simple pragmatic approach to mesh routing using batman. *2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries, CSIR*, pages 62 – 68.

Lambrecht, C. J. v. d. B. and Verscheure, O. (1996). Perceptual quality measure using a spatiotemporal model of the human visual system. In *Electronic Imaging: Science & Technology*, pages 450–461.

Lee, S., Cho, C., Hong, E.-k., and Yoon, B. (2016). Forecasting mobile broadband traffic: Application of scenario analysis and delphi method. *Expert Systems with Applications*, 44:126–137.

Schulz-Zander, J., Suresh, L., Sarrar, N., Feldmann, A., Hühn, T., and Merz, R. (2014). Programmatic orchestration of wifi networks. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 347–358. USENIX Association.

Tsukamoto, K., Yamaguchi, T., and Kashihara, S. (2007). Experimental evaluation of decision criteria for wlan handover: Signal strength and frame retransmission. *IEICE transactions on communications*, 90(12):3579–3590.

Uma Função Virtualizada de Rede para a Sincronização Consistente do Plano de Controle em Redes SDN

Giovanni V. Souza[1], **Rogério C. Turchetti**[12], **Edson T. Camargo**[13], **Elias P. Duarte Jr.**[1]

[1] Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19.081 – 81.531-980 – Curitiba – PR – Brasil

[2]CTISM - Universidade Federal de Santa Maria (UFSM)
Avenida Roraima, 1000 – 97105-900 – Santa Maria – RS – Brasil

[3]Universidade Tecnológica Federal do Paraná (UTFPR) – Toledo – PR – Brasil

gvsouza@inf.ufpr.br, turchetti@redes.ufsm.br

edson@utfpr.edu.br, elias@inf.ufpr.br

Abstract. *Switches of Software Defined Networks (SDN) are programmed by a centralized entity which is the controller. Centralized control has limited availability and scalability by definition. Distributed control strategies solve this problems, but achieving consistent synchronization across multiple SDN controllers is not a trivial endeavor. In this work we propose a VNF-Consensus, a VNF (Virtual Network Function) that implements the Paxos consensus algorithm to ensure consistency among controllers of a distributed control plane. Using this approach, controllers are decoupled from synchronization tasks and can perform their control plane activities without having to execute expensive synchronization tasks. Experimental results show the benefits of the proposed solution, in particular in terms of the optimization of resource usage and a reduction of the controller workload.*

Resumo. *A programação dos switches SDN (Software Defined Networks) nas Redes Definidas por Software é realizada por um controlador. Esse controle é centralizado, portanto tendo restrições em termos de disponibilidade e de escalabilidade. Para resolver esse problema pode-se utilizar estratégias que distribuem o plano de controle. Entretanto, implementar a sincronização entre os múltiplos controladores distribuídos não é uma tarefa trivial. O presente trabalho propõe uma solução para a sincronização consistente do plano de controle em redes SDN através da implementação de uma função virtualizada de rede (VNF – Virtual Network Function) denominada de VNF-Consensus. VNF-Consensus implementa o algoritmo Paxos e com sua utilização os controladores ficam desacoplados e podem executar em paralelo suas atividades no plano de controle. A implementação e os resultados experimentais mostram os diversos benefícios obtidos pelo uso da solução proposta, em especial a otimização no uso dos recursos computacionais e a redução na carga dos controladores.*

1. Introdução

Redes Definidas por Software (SDN – *Software Defined Networks*) têm melhorado a flexibilidade e facilitado o gerenciamento das redes de computadores. A estratégia adotada

pelas arquiteturas SDN é separar o plano de controle do plano de dados. O controle é, em geral, centralizado e responsável por tomar decisões que ocorrem no plano de dados. Nesta abordagem centralizada, o estado da rede é determinado por um controlador [Ho et al. 2016]. Considerando aspectos de gerenciamento, controle e visão global da topologia da rede, esta abordagem centralizada é, sem dúvidas, atraente. Por outro lado, a implementação do plano de controle centralizado tem, naturalmente, restrições em termos de disponibilidade, desempenho e de escalabilidade [Canini et al. 2015].

Neste sentido, há um consenso de que o plano de controle precisa ser distribuído [Schiff et al. 2016]. Além disso, para que a falha de um controlador possa ser mascarada é necessário aplicar técnicas de redundância no plano de controle [Koponen et al. 2010, Berde et al. 2014]. Entretanto, a sincronização de múltiplos controladores distribuídos não é uma tarefa trivial. Considere, por exemplo, o problema da instalação consistente de novas regras de encaminhamento de pacotes. Se regras conflitantes forem aplicadas estipulando rotas distintas haverá uma patologia na rede podendo, por exemplo, resultar em *loops* indesejados ou rotas contornando serviços de interesse.

Segundo [Canini et al. 2015] um dos principais problemas em aberto no contexto das redes SDN é justamente a necessidade de tornar o plano de controle robusto, sem interferir no desempenho global das funções oferecidas pela rede e, obviamente, preservar a correta operação do plano de dados. Entretanto, as soluções existentes para a construção de um plano de controle robusto, em geral, produzem ambientes em que deixam a cargo do controlador a função de coordenar as ações realizadas no plano de controle. Em outras palavras, são hospedados nos próprios controladores serviços que permitem *sincronizar* de maneira consistente as informações gerenciadas pelos vários controladores presentes na rede [Koponen et al. 2010, Canini et al. 2015, Ho et al. 2016]. Por outro lado, há soluções que procuram aliviar a carga dos controladores usando os próprios *switches* para sincronizar o plano de dados [Schiff et al. 2016, Dang et al. 2015]. Entretanto, estas são soluções que, em geral, implicam em alterações do funcionamento padrão, como exemplo, mudanças no protocolo *OpenFlow*.

Portanto, acredita-se que aumentar as múltiplas tarefas que os controladores já exercem na rede, não é a estratégia mais adequada, uma vez que a inclusão dessas funcionalidades extras implica no aumento da carga de trabalho. Além disso, segundo os autores em [Karakus and Durresi 2017] problemas quanto ao desempenho do plano de controle ainda é um assunto que precisa ser investigado tanto pela comunidade acadêmica quanto pela indústria. Sendo assim, o presente trabalho propõe uma solução para a sincronização do plano de controle em redes SDN, onde os controladores são coordenados por uma função virtualizada de rede (VNF – *Virtual Network Function*). A VNF proposta é denominada de *VNF-Consensus* e implementa o algoritmo de consenso Paxos [Lamport 1998] no ambiente de rede. Dessa forma, a *VNF-Consensus* consegue manter um plano de controle consistente pois sincroniza as ações entre todos os controladores SDN. Conceitualmente, uma VNF é a representação de dispositivos de redes em entidades virtuais que são executadas utilizando tecnologias de virtualização baseadas em software [ETSI 2016].

Para obter uma visão comum do estado da rede, cada controlador possui acesso a uma instância da *VNF-Consensus* através da qual poderá receber decisões e enviar dados para serem sincronizados. Dessa maneira, todas as decisões executadas pela *VNF-*

Consensus são sistematicamente executadas sem a atuação direta dos controladores. As vantagens dessa abordagem são: (i) os controladores ficam desacoplados e podem executar em paralelo suas atividades no plano de controle; (ii) o plano de controle é sincronizado sem aumentar a carga de trabalho dos controladores, pois a *VNF-Consensus* é executada como entidade externa aos controladores, não utilizando os mesmos recursos computacionais; (iii) como os controladores não participam diretamente nas decisões do Paxos, o consenso consegue garantir, independentemente do número de controladores operacionais, a conclusão de seus serviços; (iv) por fim, a solução proposta não exige nenhuma alteração no funcionamento padrão do protocolo *OpenFlow* ou nos *switches* SDN. Resultados experimentais mostram o desempenho da *VNF-Consensus* e seu impacto na utilização dos recursos, como também os benefícios obtidos por executar o serviço sem a participação direta dos controladores SDN.

O restante deste trabalho está organizado da seguinte forma. A seção 2 apresenta trabalhos relacionados que tratam da sincronização do plano de controle em redes SDN. Na seção 3 é apresentada a proposta focando no aspecto de implementação e no algoritmo de consenso. Os experimentos e as conclusões do trabalho são apresentados nas seções 4 e 5, respectivamente.

2. Sincronização dos Dados em Redes SDN

Nesta seção apresentamos os trabalhos relacionados que tratam da sincronização do plano de controle em redes SDN. A principal diferença desses trabalhos para a solução proposta é que no presente trabalho há um esforço em permitir a sincronização consistente no plano de controle através do uso de uma função virtualizada de rede.

Para garantir a consistência nas operações de rede, as ações executadas em diferentes controladores SDN precisam ser sincronizadas. Neste contexto, em [Schiff et al. 2016] os autores propõem um *framework* para a sincronização das informações no plano de dados implementadas dentro (*in-band*) dos *switches*. Segundo os autores, protocolos convencionais que executam suas funções fora dos *switches* possuem um alto custo para coordenação e sincronização das informações. Em contraste, soluções *in-band* permitem resolver problemas de acordo (*agreement*) através da troca de poucas mensagens. A solução é baseada no método 'Compare-And-Set' (CAS) abordagem utilizada para a consistência de memória. Esse método garante atomicidade nas transações evitando cenários inconsistentes. Em outras palavras, para evitar inconsistência nas regras instaladas nos *switches* pelos controladores, o comando *FlowMod* é modificado e usa *flags* para definir quais regras devem ser adicionadas ou removidas. Os autores apresentam uma prova de conceito na qual demonstram a eficiência da solução proposta. O mecanismo apresentado pelos autores é simples e pode ser implementado sem a necessidade de hardware ou protocolos extras. Por outro lado, para que os algoritmos propostos funcionem, há a necessidade de efetuar alterações nas *flags* do protocolo *OpenFlow*.

Em [Dang et al. 2015] os autores propõem mover a lógica do algoritmo de consenso Paxos para ser executado dentro da rede. Em particular, propõem que o algoritmo seja executado nos próprios *switches* SDN. Duas abordagens são propostas: a implementação do algoritmo Paxos na sua versão completa, isto é, sem otimizações, para ser executado nos *switches* SDN; e a execução de uma versão otimizada denominada de NetPaxos que evita a implementação de um coordenador (elemento do Paxos) entre os *switches*. Os

autores mostram que mover a lógica do consenso para dentro da rede reduz a complexidade das aplicações, reduz a latência das mensagens da aplicação e aumenta o *throughput* das transações. Entretanto, os *switches* precisam executar e assumir papéis que são atribuídos pelo algoritmo Paxos, essa abordagem dificulta sua execução pois exige mudanças no funcionamento padrão da rede, incluindo a alteração do *firmware* nos *switches* SDN.

Em [Ho et al. 2016] os autores propõem uma solução para a sincronização e consistência das informações que são gerenciadas pelos controladores em uma rede SDN. O objetivo é manter a consistência entre múltiplos controladores através da implementação de um algoritmo de consenso. Uma versão ao algoritmo Paxos denominada de FPC (*Fast Paxos-based Consensus*) é proposta. Segundo os autores, o algoritmo FPC reduz a complexidade se comparado ao algoritmo original proposto por Lamport (1998). Em especial, FPC não possui um líder pré-definido, ao invés disso o algoritmo permite que qualquer processo participante possa se tornar o líder (denominado de *chairman*). Uma vez que todos os processos (controladores) tenham realizado a atualização, o *chairman* retorna a seu papel de *listener* terminando a rodada do algoritmo. Os autores comparam o FPC com o algoritmo de consenso Raft [Ongaro and Ousterhout 2014] e concluem que o algoritmo proposto apresenta menor tempo para a execução do consenso.

Onix [Koponen et al. 2010] é uma plataforma que permite a implementação do plano de controle como um sistema distribuído mantendo uma visão global da rede. Onix é um sistema distribuído executado em um *cluster* de servidores físicos. No Onix um controlador armazena informações em uma estrutura de dados denominada de NIB (*Network Information Base*). A NIB representa a parte mais importante da plataforma, pois todo o estado da rede é sincronizado através de leituras e escritas controladas pela base de dados. Em especial, Onix representa um conjunto de APIs que fornecem escalabilidade e confiabilidade por replicar e distribuir as informações sincronizadas entre múltiplas instâncias na rede. Se uma informação for alterada, a mesma será propagada por todas as instâncias de NIBs garantindo a consistência através da coordenação de um algoritmo de consenso (Zookeeper [Hunt et al. 2010]). Dito pelos próprios autores, Onix não é uma inovação em si, pois derivam de diversos trabalhos propostos ao longo da história.

Em [Canini et al. 2015], os autores propõem um modelo formal para a comunicação entre o plano de dados e o plano de controle distribuído de uma rede SDN. Em particular, os autores desejam tratar o problema de inconsistência durante a atualização de regras que são instaladas em um ou mais *switches*. Estas regras são as políticas da rede e a solução deve, informalmente, garantir que um pacote que está trafegando na rede seja processado por exatamente uma única política, mesmo em situações em que ela (a política) eventualmente seja atualizada. Os autores apresentam um protocolo denominado de *ReuseTag* que usa uma abordagem de máquina de estados para implementar a ordem total das atualizações das políticas, a solução assume que o plano de dados é como se fosse uma estrutura de memória compartilhada onde os controladores são os processos que fazem leituras e escritas nessa memória. Dado um limite máximo de latência na rede e assumindo um número máximo de f processos/controladores falhos, o protocolo *ReuseTag* funciona corretamente. Por fim, são definidos ainda requisitos mínimos para a sincronização das ações entre os controladores demonstrando a complexidade dessa sincronização mesmo na presença de um algoritmo de consenso.

Tabela 1. Comparação dos trabalhos com relação a sincronização dos dados.

Trabalhos Relacionados	Algoritmo de Sincronização	Plano de Sincronização	Local de Sincronização dos Algoritmos
[Schiff et al. 2016]	Transações atômicas usando *compare-and-set* (CAS)	Plano de Dados	Switches SDN
[Dang et al. 2015]	Algoritmo de consenso *NetPaxos*	Plano de Dados	Switches SND
[Ho et al. 2016]	Algoritmo de consenso *Fast Paxos-based Consensus*	Plano de Controle	Controladores SND
[Koponen et al. 2010]	Algoritmo de consenso *Zookeeper*	Plano de Controle	Controladores SDN
[Canini et al. 2015]	Algoritmo *Policy Serialization* (PS) baseado em Rep. de Máquina de Estados	Plano de Controle	Controladores SDN
Plataforma ODL	Algoritmo de consenso Raft	Plano de Controle	Controladores SDN

O OpenDaylight[1] (ODL) é um controlador de rede SDN que oferece em sua arquitetura diversas funcionalidades, em especial a possibilidade de executar e de sincronizar múltiplos controladores SDN. Essa arquitetura é denominada de ODL *Clustering* e oferece um mecanismo de integração entre múltiplos processos e aplicações. Cada controlador possui seu próprio local de armazenamento e a replicação dos dados ocorre através de caches distribuídas aplicando o esquema Infinispan[2]. Toda a comunicação e notificação entre os controladores no ODL *Clustering* ocorre usando um grupo de ferramentas denominado de *Akka*. Essa comunicação ocorre somente entre os controladores da rede, as decisões e a consistência são garantidas pelo uso do protocolo de consenso Raft. O Raft é utilizado para coordenar as atualizações que são executadas entre os controladores SDN. Note que toda a sincronização das informações exige a execução de algoritmos que estão plugados diretamente em cada controlador. Como resultado há um esforço adicional executado pelos controladores a fim de obter a consistência das informações na rede.

A Tabela 1 apresenta os trabalhos relacionados nesta seção descrevendo os mecanismos de sincronização utilizados por cada solução. Vale ressaltar que a principal diferença no presente trabalho pode ser visualizada na coluna que especifica o local de sincronização. Note que a proposta neste trabalho executa a sincronização no plano de controle sem a participação direta das entidades da rede SDN, como exemplo os *switches* e os controladores. Mais detalhes da abordagem utilizada são apresentadas na próxima seção.

3. Um Plano de Controle Robusto Baseado em VNF

Nesta seção é apresentada uma caracterização do problema abordado: a consistência do plano de controle em redes SDN. Na sequência é detalhada a arquitetura e as particularidades de implementação, bem como, uma breve descrição do algoritmo de consenso Paxos e detalhes de como sua lógica é implementada e executada na *VNF-Consensus*.

3.1. Caracterização e Delimitação do Problema

Para garantir um plano de controle consistente em uma rede SDN, todos os eventos a serem atualizados precisam ser sincronizados entre os controladores envolvidos. Cada

[1]https://www.opendaylight.org/
[2]http://infinispan.org/

atualização gera um novo estado da rede. As mudanças de estados são causadas por eventos como, por exemplo: a criação de um novo fluxo de comunicação, a ocorrência de falhas em *links* ou *switches*, a instalação de regras para passar por serviços como filtros de pacotes ou novas rotas, entre outros. Em síntese, neste contexto é necessário utilizar algum mecanismo que sincronize cada evento evitando estados inconsistentes. Os autores de [Schiff et al. 2016] descrevem diversas patologias causadas na rede devido a esses estados inconsistentes.

As decisões realizadas no plano de controle implicam em mudanças no plano de dados, isto é, a criação de um novo fluxo de dados implicará na instalação de uma nova regra no *switch*. Portanto, é importante entender a comunicação entre os controladores e os *switches* SDN. Em [Tianzhu et al. 2016] os autores identificam dois tipos de comunicação (ver Figura 1): *Switch-to-Controller* e *Controller-to-Controller*.

A comunicação *Switch-to-Controller* suporta iterações entre os *switches* e o controlador usando o protocolo *OpenFlow*. Por exemplo, é nesse plano que um pacote *packet-in* (pacote gerado quando não há uma correspondência na tabela de fluxos do *switch*) é encaminhado do *switch* ao controlador. O controlador, por sua vez, retorna uma mensagem *flow-mod* para autorizar ou não a comunicação do pacote. Essas são mensagens utilizadas para gerenciar as tabelas de fluxos de cada *switch*. Considerando a execução de múltiplos controladores, a decisão tomada para a autorização do novo fluxo de pacotes deve ser sincronizada entre todos os controladores da rede. Caso contrário, estados inconsistentes no plano de controle poderão causar patologias na rede, como o descarte de pacotes, criação de *loops*, rotas contornando serviços específicos, entre outros. Além disso, a mensagem *flow-mod* precisa ser atualizada pelos *switches* no mesmo instante de tempo (no contexto do plano de dados), evitando estados inconsistentes. Entretanto, esta tarefa pode ser significativamente complicada em ambientes onde os atrasos não podem ser previstos [Zhou et al. 2014]. Portanto, vale ressaltar que neste trabalho abordamos a consistência no plano de controle, ao passo que a sincronização no plano de dados pode ocorrer em instantes diferentes de tempo. Em resumo, assumimos que nos estados transientes do plano de dados há a possibilidade de ocorrer períodos com estados inconsistentes da rede.

A comunicação *Controller-to-Controller* permite a sincronização do plano de controle. A consistência nesse plano exige que todos os controladores possuam a mesma visão do estado da rede. No plano de controle a consistência pode ser forte ou eventual [Tianzhu et al. 2016]. A consistência forte significa que as leituras dos dados em diferentes controladores produzem sempre o mesmo resultado, ao passo que, a consistência eventual significa que há períodos transientes em que as leituras em diferentes controladores podem produzir diferentes valores. No presente trabalho utilizamos o algoritmo de consenso Paxos que implementa a consistência forte.

3.2. VNF-Consensus

Em uma rede SDN todas as ações executadas por um conjunto de controladores precisam ser sincronizadas e coordenadas. A *VNF-Consensus* implementa um algoritmo de consenso que possibilita garantir a consistência do plano de controle em redes SDN. Em particular, a *VNF-Consensus* implementa, no ambiente de rede, o algoritmo de consenso Paxos descrito a seguir. Note que, embora sucinta, a descrição do algoritmo é importante para a compreensão da *VNF-Consensus* que será detalhada na sequência.

Paxos é um algoritmo de consenso tolerante a falhas proposto por Leslie Lamport [Lamport 1998]. Informalmente, um algoritmo de consenso [Santos et al. 2011] permite que um conjunto de processos entre em acordo sobre um determinado valor, sendo que inicialmente cada processo pode propor valores distintos. Dentre as propriedades que um algoritmo de consenso deve atender, destaca-se neste trabalho duas propriedades: segurança (*safety*) e progresso (*liveness*). A segurança garante que todos os processos sem falha decidem pelo mesmo valor. O progresso garante que o consenso termina com sucesso. No algoritmo Paxos os processos podem assumir três diferentes papéis denominados de: *proposers*, *acceptors* e *learners*.

Os *proposers* propõem um valor, os *acceptors* escolhem um valor e os *learners* aprendem o valor decidido. Um único processo pode assumir qualquer uma dessas funções e múltiplas funções simultaneamente. Paxos é ótimo em termos de resiliência [Lamport 2006]: para tolerar f falhas ele requer $2f + 1$ *acceptors* – isto é, para assegurar o progresso um quórum de $f + 1$ *acceptors* devem estar sem-falha. Para resolver o consenso, Paxos executa em rodadas. Tipicamente um *proposer* ou um *acceptor* atua como *coordenador* da rodada. O *coordenador* recebe propostas que tentam alcançar o consenso. Quando o quórum de *acceptors* aceita o mesmo número de rodada, o consenso termina. Neste momento os *learners* têm acesso ao valor decidido.

Paxos é utilizado neste trabalho para sincronizar os dados no plano de controle. Em especial, os controladores aprendem (*learners*) os valores decididos e fornecidos pela *VNF-Consensus*. Dessa maneira, o plano de controle mantém o estado da rede em todos os controladores. Paxos é baseado em um modelo de consistência forte. Portanto, a *VNF-Consensus* garante que os dados resultantes das leituras de diferentes controladores irão produzir sempre o mesmo resultado.

Para obter uma visão comum do estado da rede, cada controlador possui acesso a uma instância da *VNF-Consensus* através da qual poderá receber decisões e enviar dados para serem sincronizados. A Figura 1 apresenta a integração dos controladores SDN com a *VNF-Consensus*. Todas as decisões realizadas pela *VNF-Consensus* são sistematicamente executadas sem a atuação direta dos controladores. Assim, cada controlador é um cliente do algoritmo de consenso que irá aprender os resultados gerados pela *VNF-Consensus*.

Considere que o *host* x apresentado na Figura 1 deseja, pela primeira vez, fazer uma comunicação com outro dispositivo qualquer. O *host* x encaminha o pacote ao *switch OpenFlow* que, por sua vez, verifica se há regras deste destinatário em sua tabela de fluxos. Se nenhuma entrada existir na tabela de fluxos referente ao destinatário, o *switch* encaminha um *packet-in* ao controlador SDN. Antes do controlador decidir o que fazer com o pacote, ele encaminha uma cópia do mesmo para a *VNF-Consensus*. O consenso então é inicializado e, ao final, o resultado é encaminhado a todos os controladores SDN via comunicação REST (*REpresentational State Transfer*).

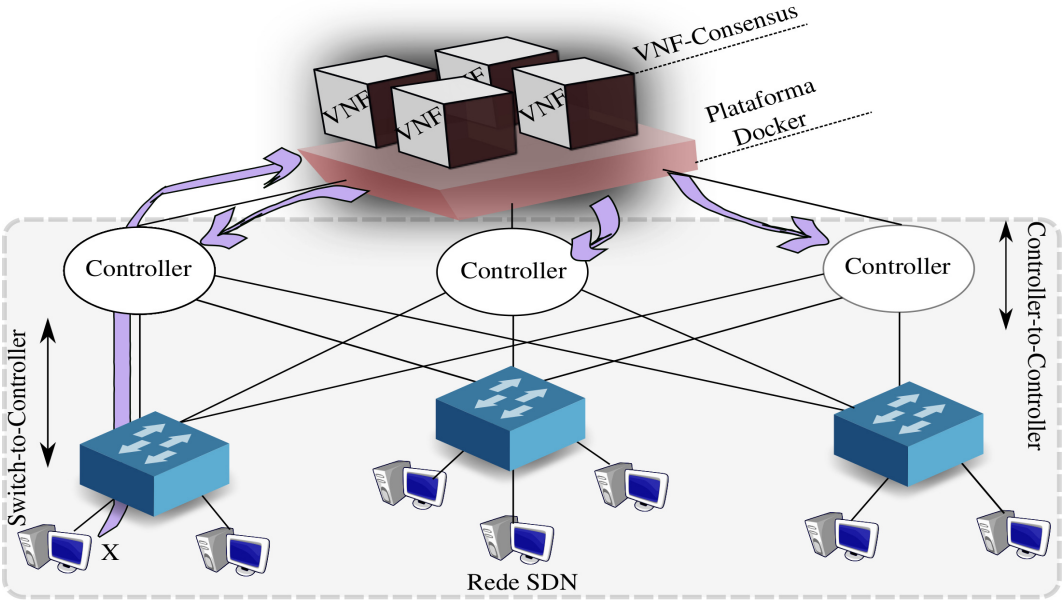


Figura 1. Integração da VNF na arquitetura do controlador SDN.

Observe que quando o controlador SDN recebe um novo fluxo, como no exemplo apresentado no parágrafo anterior, faz-se necessário executar um 'desvio' no fluxo dos pacotes com o objetivo de encaminhá-los à *VNF-Consensus*. Em outras palavras, cada pacote precisa ser classificado. A classificação permite definir qual pacote será encaminhado e para qual destino. Se for decidido que um pacote precisa ser encaminhado para a *VNF-Consensus*, então um fluxo de encaminhamento precisa ser criado.

Para executar essa tarefa é proposta uma ferramenta denominada de *Filter*. A *Filter* realiza a classificação e, quando necessário, o encaminhamento dos pacotes para a *VNF-Consensus*. O funcionamento e a integração desses componentes podem ser visualizados na arquitetura apresentada na Figura 2. Observe que as instâncias da *VNF-Consensus* são executadas em *containers* no ambiente Docker[3].

Note (na arquitetura apresentada na Figura 2), que a ferramenta *Filter* é composta por dois módulos denominados de *Classifier* e *Forwarder*. O *Classifier* possui acesso às regras que são armazenadas no repositório *Rules*. Essas regras permitem classificar os fluxos de entrada para determinar o destino de cada pacote. O processo ocorre de acordo com os passos mostrados na Figura 2 e detalhados a seguir: o controlador SDN encaminha todos os pacotes para a ferramenta *Filter* (passo 1), se houver alguma regra cadastrada no módulo *Classifier* que se encaixe ao pacote recebido (passo 2), o pacote é repassado para o módulo *Forwarder* (passo 4), caso contrário, o pacote será retornado ao controlador (passo 3). O *Forwarder*, por sua vez, é responsável por realizar o repasse dos pacotes, por exemplo, permitindo encaminhar o pacote à correspondente VNF (passo 5). Esse último passo também é identificado como SFP (*Service Function Path*)

[3]http://www.docker.org/

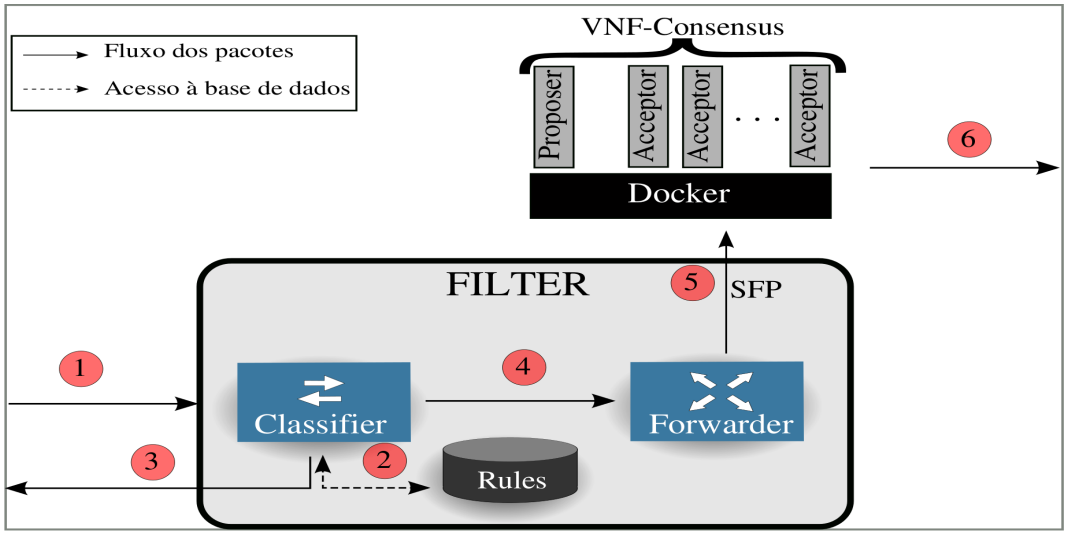


Figura 2. Arquitetura para a sincronização do plano de controle.

Quando um SFP é criado, o pacote contendo a regra a ser instalada é repassado para o coordenador da *VNF-Consensus*. Do ponto de vista do Paxos, neste cenário cada VNF executa funções distintas de acordo com as especificações do algoritmo. Isto é, na arquitetura apresentada na Figura 2 considere o *proposer* como sendo também o coordenador. É o coordenador quem recebe as regras vindas dos diversos controladores da rede. Ao receber uma regra, o coordenador então inicia uma nova instância de consenso. Uma instância é definida executando duas fases distintas do algoritmo Paxos. Na primeira fase, o coordenador seleciona um número único para a rodada e o envia em uma solicitação para os *acceptors*. Ao receber a solicitação com um número maior que qualquer outro recebido previamente para aquela instância, o *acceptor* responde ao coordenador prometendo que rejeitará qualquer solicitação futura com um número de rodada menor. Se o *acceptor* já aceitou uma regra para a instância, ele retorna essa regra ao coordenador junto com o número de rodada recebido quando a regra foi aceita. Quando o coordenador recebe respostas de um quórum de *acceptors*, a segunda fase do protocolo é inicializada.

Na segunda fase, o coordenador verifica os valores recebidos do quórum de *acceptors*. Se não houver *acceptors* que tenham aceitado uma regra, o coordenador executa a fase 2 com sua regra. Por outro lado, se algum *acceptor* retornou uma regra na primeira fase, o coordenador é obrigado a executar a segunda fase com a regra que possuir o maior número de rodada. Com a regra selecionada, o coordenador envia uma requisição aos *acceptors* com o número da rodada e a regra escolhida. Se os *acceptors* não prometeram participar de uma rodada com número maior, então eles respondem ao coordenador aceitando a regra proposta. Quando o coordenador recebe respostas de um quórum de *acceptors* há a garantia de que não há outro *proposer*, ou coordenador, com um número de proposta maior que a já aceita. A regra então é enviada aos *learners* e o consenso para aquela instância termina.

O coordenador envia o valor decidido aos *learners* que são controladores SDN que, dessa forma, aprendem o valor decidido pela *VNF-Consensus*.

Note que para cada papel definido no algoritmo Paxos um *container* é criado para

ser executado de forma independente entre os processos. Essa abordagem permite o isolamento total dos processos. A implementação em *containers* é motivada por sua baixa utilização dos recursos computacionais, rápida instanciação e fácil implementação de funções virtualizadas, possibilitando também uma alta densidade de processos virtualizados no mesmo *host* [Anderson et al. 2016]. A seguir são apresentados os experimentos que permitem comprovar os benefícios do serviço descrito nesta seção.

4. Avaliação da NFV-Consensus

Nesta seção são apresentados experimentos executados para avaliar o desempenho da *VNF-Consensus* em uma rede SDN. O ambiente para execução dos experimentos é implementado usando o protocolo *OpenFlow* e controladores *Ryu*[4]. O ambiente é hospedado em uma máquina física com as seguintes características: processador AMD FX-4300 CPU 3.8GHz com 4 núcleos e sistema operacional *Ubuntu* 16.04 com *kernel* 4.4.0-53. Para os experimentos apresentados nesta seção, a rede é virtualizada através da ferramenta *Mininet*[5].

A topologia consiste de 3 *switches*, 3 controladores remotos e 3 *hosts* que são utilizados para gerar pacotes para a sincronização do plano de controle. A *VNF-Consensus* é configurada com 1 *proposer*, 3 *acceptors* e 3 *learners*, estes últimos são os controladores SDN. Além disso, o fluxo de dados foi gerado a partir do *cbench*[6], uma ferramenta que tem como objetivo sobrecarregar o controlador através de múltiplos pacotes do tipo *packet-in*.

A seguir são apresentados três grupos de experimentos. No primeiro grupo o objetivo é avaliar e comparar o impacto causado para a sincronização do plano de controle. No segundo grupo comparamos o tempo para inicialização das instâncias do Paxos. Por fim, no terceiro grupo é avaliado e comparado o *throughput* para a execução do Paxos.

Custo para a sincronização do plano de controle

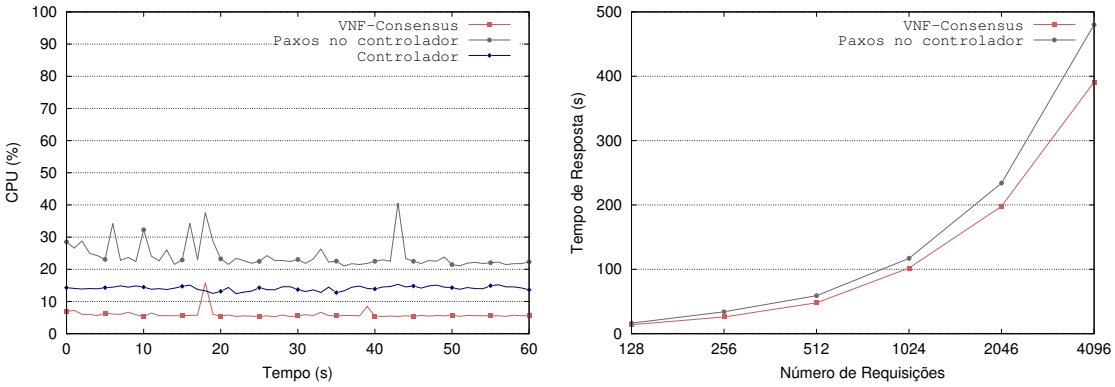
Quando a sincronização do plano de controle é deixada a cargo dos próprios controladores SDN, aumenta-se a quantidade de tarefas que os controladores precisam executar. Nestas condições, pode-se ter um impacto negativo no desempenho global da rede. Para medir tal impacto, a figura 3 apresenta os seguintes experimentos: a utilização de CPU e o tempo para um controlador instalar um conjunto de regras.

O objetivo do experimento apresentado na figura 3(a) é avaliar o custo para a sincronização do plano de controle em termos de utilização de CPU. Em particular, deseja-se medir a carga extra de trabalho causada pelas ações de sincronização realizadas pelos próprios controladores. Para este experimento, os controladores foram submetidos a um fluxo de dados gerado a partir dos *hosts* através da ferramenta *cbench*. Note que a *VNF-Consensus* apresenta em média 5,9% de utilização de CPU. Quando os controladores não sincronizam o plano de controle, a utilização média de CPU é de aproximadamente 14.1%. Porém, quando o Paxos é executado nos controladores há um impacto maior na utilização de CPU, atingindo aproximadamente 24,1%. Essa sobrecarga dos controladores é causada pelas ações de sincronização do plano de controle somada ao processamento

[4]https://osrg.github.io/ryu/

[5]http://mininet.org/

[6]https://github.com/mininet/oflops/tree/master/cbench



(a) Uso de CPU.

(b) Tempo necessário para instalar um conjunto de regras.

Figura 3. Comparação quanto ao impacto causado para a sincronização do plano de controle: execução dentro e fora dos controladores.

dos novos fluxos gerados na rede. Como resultado, há um impacto negativo no desempenho global da rede. Esse cenário pode ser melhor visualizado nos experimentos da figura 3(b).

No experimento da figura 3(b) o objetivo é verificar se os controladores apresentam algum impacto negativo no processamento normal da rede quando eles também precisam sincronizar o estado da rede. Para o experimento foi utilizado um *script* que cria 128 processos que produzem, de maneira contínua, requisições REST que instalam regras aleatórias no *switch*. Em paralelo, a ferramenta *cbench* envia um fluxo de dados fazendo com que o plano de controle seja constantemente sincronizado. Nestas condições, avaliamos o tempo necessário para instalar um conjunto de regras. Vale ressaltar que esse tempo é medido desde o início da requisição até o seu respectivo retorno, isto é, quando o *switch* instala a regra e retorna uma confirmação ao controlador.

Note que conforme aumenta-se o número de requisições, o tempo de resposta aumenta em ambos os casos. Entretanto, quando a sincronização é realizada pela *VNF-Consensus*, observa-se uma redução de até 18,5% no tempo para instalação de um conjunto de regras. Como mostrado na figura 3(b), essa diferença tende a crescer conforme o número de requisições aumenta. Claramente podemos notar os benefícios da sincronização do plano de controle via *VNF-Consensus*.

Tempo para inicializar uma instância do Paxos

Conforme visto na seção 3.2, o Paxos é ótimo em termos de resiliência, já que requer $2f + 1$ *acceptors* para tolerar f falhas. Para aumentar o grau de resiliência é necessário aumentar o número de participantes. Na abordagem onde os próprios controladores executam a sincronização através do Paxos, para aumentar o número de participantes é necessário aumentar o número de controladores. Essa abordagem impõe restrições, tendo em vista que a inclusão de novos controladores exige a inserção de uma infraestrutura subjacente (e.g., mecanismo de virtualização, armazenamento, etc.). Por outro lado, ao utilizar o Paxos como uma VNF, para aumentar a resiliência, basta criar novas instâncias

Tabela 2. Paxos instanciado via *container* na *VNF-Consensus* vs. Paxos instanciado via VM no controlador.

Sistema de Virtualização	Média (s)	Máximo (s)	Mínimo (s)	Desvio Padrão (s)
Paxos *VNF-Consensus*	1.73	2.04	1.58	0.15
Paxos controlador	21.02	23.09	20.18	1.02

da *VNF-Consensus*. Em outras palavras, bastaria criar um novo *container* e iniciar uma nova instância da *VNF-Consensus*.

Concretamente, neste experimento comparamos o tempo para iniciar uma nova instância da *VNF-Consensus* versus o tempo para iniciar um novo controlador SDN. Para avaliar tal custo, o experimento apresentado na tabela 2 mostra a diferença de tempo para iniciar uma instância da *VNF-Consensus* e iniciar um controlador SDN, ambos executando o Paxos. Vale ressaltar que os controladores estão hospedados em máquinas virtuais via *Virtualbox*[7]. Os dados apresentados são resultados de 10 execuções e a medição de tempo foi realizada através do comando *time* do próprio *Linux*.

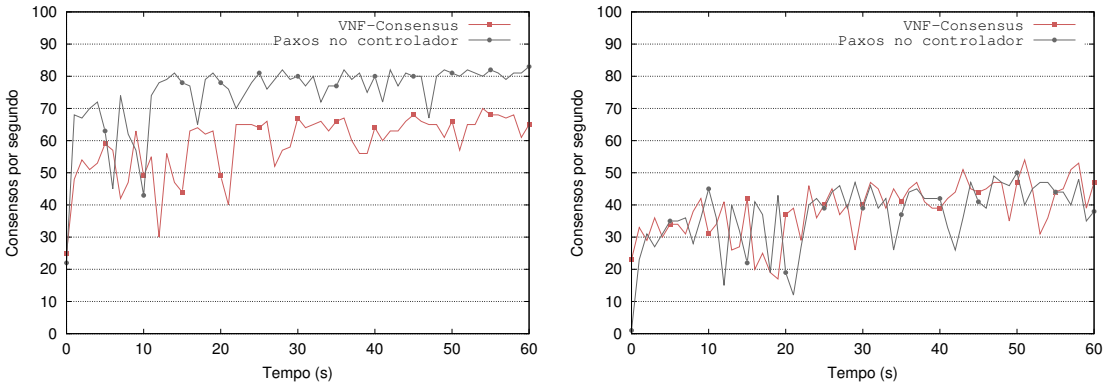
Na tabela 2 podemos observar que o tempo para inicializar uma nova instância via *container* é menor do que via máquina virtual, na comparação entre os valores médios há uma redução de aproximadamente 88% do tempo para instanciar o Paxos via *container*. Além disso, ao analisar os valores máximo, mínimo e desvio padrão podemos notar que a implementação via *container* apresenta baixa variação. Os resultados demonstram claramente a vantagem de executar a função de sincronização como uma VNF, ao invés de acrescentar esta funcionalidade ao controlador.

Análise do throughput do Paxos

Os experimentos apresentados na figura 4 tem como objetivo mostrar e comparar o *throughput* do Paxos, isto é, o número de execuções por segundo quando o consenso está sendo executado na *VNF-Consensus* e quando ele está hospedado nos controladores. No experimento da figura 4(a) o controlador foi submetido a um fluxo de dados com requisições contínuas de atualizações para forçar a sincronização do plano de controle. É possível notar que quando o consenso está sendo executado no controlador há um aumento no *throughput* se comparado à *VNF-Consensus*. Esse resultado é, de certa forma, previsível, uma vez que as instâncias da VNF estão localizadas fora dos controladores e, para toda execução do Paxos, há a necessidade de comunicação entre as VNFs e os controladores. Isso onera o tempo de execução do Paxos via *VNF-Consensus*.

No experimento da figura 4(b), além do fluxo de dados gerado pela ferramenta *cbench*, o controlador é sobrecarregado com requisições REST, provenientes do mesmo *script* utilizado no experimento da figura 3(b). As requisições em paralelo forçam os controladores a executarem consultas contínuas às tabelas de fluxos. Como resultado, note que ambos os casos apresentam uma redução no *throughput*. Porém, quando o Paxos está sendo executado nos controladores a redução foi de aproximadamente 53,3%, ao passo que na *VNF-Consensus* essa redução foi menor, isto é, de aproximado 38,8%. Po-

[7] www.virtualbox.org



(a) Os controladores não executam atividades em paralelo.

(b) Os controladores são forçados a executarem atividades em paralelo.

Figura 4. Comparação quanto ao throughput do Paxos.

demos concluir que apesar da *VNF-Consensus* sofrer com a sobrecarga dos controladores, o impacto maior ocorre quando Paxos está sendo executado nos controladores.

Por fim, ao analisarmos todos os resultados realizados neste trabalho podemos concluir que quando a sincronização do plano de controle é realizada pela *VNF-Consensus*, tem-se uma redução na carga dos controladores, o que tem impacto significativo na escalabilidade, na medida em que há um claro limite para a quantidade de tarefas que um controlador pode executar antes de se tornar um gargalo da rede.

5. Conclusão

Neste trabalho foi proposta uma solução para a sincronização do plano de controle em redes SDN, na qual um grupo de controladores distribuídos se mantêm consistentes com o auxílio de uma função virtualizada de rede denominada de *VNF-Consensus*. A *VNF-Consensus* implementa o algoritmo de consenso Paxos no ambiente de rede. Dessa forma, a *VNF-Consensus* consegue manter um plano de controle consistente pois sincroniza as ações entre todos os controladores envolvidos. Além disso, todas as decisões realizadas pela *VNF-Consensus* são executadas sem a atuação direta dos controladores.

Os resultados experimentais mostraram que quando os próprios controladores executam a sincronização do plano de controle, há um impacto negativo no desempenho global da rede. Por outro lado, com a utilização da *VNF-Consensus*, o plano de controle é sincronizado sem aumentar a carga de trabalho nos controladores. Por consequência, vimos que há melhorias no desempenho da rede. Outro benefício da *VNF-Consensus* refere-se a resiliência em que o Paxos consegue garantir, independentemente do número de controladores operacionais, a conclusão de seus serviços, isto é, a consistência dos controladores. Além disso, a implementação da *VNF-Consensus* em *containers*, como proposto neste trabalho, mostrou ter benefícios, em especial, a otimização no uso dos recursos computacionais, como também rápida instanciação dos serviços.

Para trabalho futuro planeja-se a implementação de um serviço para a sincronização do plano de dados de uma rede SDN, isto é, visando manter a consistência entre todos

os *switches* da rede.

Referências

[Anderson et al. 2016] Anderson, J., Hu, H., Agarwal, U., Lowery, C., Li, H., and Apon, A. (2016). Performance considerations of network functions virtualization using containers. In *International Conference on Computing, Networking and Communications (ICNC'16)*.

[Berde et al. 2014] Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., Lantz, B., O'Connor, B., Radoslavov, P., Snow, W., and Parulkar, G. (2014). ONOS: Towards an Open, Distributed SDN OS. In *3th Workshop on Hot Topics in Software Defined Networking (HotSDN)*.

[Canini et al. 2015] Canini, M., Kuznetsov, P., Levin, D., and Schmid, S. (2015). A distributed and robust SDN control plane for transactional network updates. In *IEEE Conference on Computer Communications (INFOCOM)*.

[Dang et al. 2015] Dang, H. T., Sciascia, D., Canini, M., Pedone, F., and Soulé, R. (2015). Netpaxos: Consensus at network speed. In *Symposium on Software Defined Networking ResearchSymposium on Software Defined Networking Research, (SOSR'15/SIGCOMM)*.

[ETSI 2016] ETSI (Available at http://www.etsi.org/technologies-clusters/technologies/nfv, Accessed on October 02, 2016). Etsi gs nfv 002: Architectural framework.

[Ho et al. 2016] Ho, C. C., Wang, K., and Hsu, Y. H. (2016). A fast consensus algorithm for multiple controllers in software-defined networks. In *18th International Conference on Advanced Communication Technology (ICACT)*.

[Hunt et al. 2010] Hunt, P., Konar, M., Junqueira, F. P., and Reed, B. (2010). ZooKeeper: Wait-free Coordination for Internet-scale Systems. In *USENIX Conference on USENIX Annual Technical Conference (USENIXATC)*.

[Karakus and Durresi 2017] Karakus, M. and Durresi, A. (2017). A survey: Control plane scalability issues and approaches in software-defined networking (SDN). *Computer Networks*, 112.

[Koponen et al. 2010] Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., and Shenker, S. (2010). Onix: A Distributed Control Platform for Large-scale Production Networks. In *9th Conference on Operating Systems Design and Implementation (OSDI)*.

[Lamport 1998] Lamport, L. (1998). The Part-time Parliament. *ACM Transactions on Computer Systems (TOCS)*, 16(2).

[Lamport 2006] Lamport, L. (2006). Lower bounds for asynchronous consensus. *Distributed Computing*, 19(2).

[Ongaro and Ousterhout 2014] Ongaro, D. and Ousterhout, J. (2014). In Search of an Understandable Consensus Algorithm. In *USENIX Conference on USENIX Annual Technical Conference (USENIXATC)*.

[Santos et al. 2011] Santos, N., Schiper, A., Hutle, M., and Borran, F. (2011). Quantitative analysis of consensus algorithms. *IEEE Transactions on Dependable and Secure Computing*, 9.

[Schiff et al. 2016] Schiff, L., Schmid, S., and Kuznetsov, P. (2016). In-Band Synchronization for Distributed SDN Control Planes. *SIGCOMM Comput. Commun. Rev.*, 46(1).

[Tianzhu et al. 2016] Tianzhu, Z., Andrea, B., Samuele De, D., and Paolo, G. (2016). The role of inter-controller traffic for placement of distributed sdn controllers. In *Networking and Internet Architecture.*

[Zhou et al. 2014] Zhou, B., Wu, C., Hong, X., and Jiang, M. (2014). Programming network via distributed control in software-defined networks. In *2014 IEEE International Conference on Communications (ICC)*.

Multi Controllers Architecture with Adaptive Monitoring in SDWMN In-Band

Bruno Ramos e Silva[1], **Madson R. Araujo**[1], **Ibirisol F. Ferreira**[1], **Gustavo B. Figueiredo**[1]

[1]Institute of Mathematics and Statistics – Federal University of Bahia (UFBA)
Av. Adhemar de Barros, s/n, Ondina – 40170-115 – Salvador – BA – Brasil

{brunoramos,madsonra,ibirisol,gustavo}@dcc.ufba.br

Abstract. *Software Defined Wireless Mesh Networks (SDWMN) enable more intelligent and programmable wireless networks. SDWMN with in-band control plane can optimize resource utilization when compared to out-of-band network architectures, because in-band can exempt the need of additional physical or virtual interfaces and legacy protocols to coordinate the control plane traffic. This traffic can also be engineered by SDN and make use of its possible benefits. However, there are many challenges to be dealt with SDWMN in-band, such as network auto-configuration during start-up, scalability and control plane overhead over the data plane. To overcome these limitations, we propose in this paper a multi controllers architecture with adaptive monitoring in SDWMN in-band which objective is to reduce control plane overhead and delay while improving network scalability. The work is implemented as a proof-of-concept in the OpenWiMesh framework. Results show it effectively reduce switch-controller latency and control plane overhead, as well as enabling higher scalability. We also show our monitoring solution enables better wireless network view and uses at least near half the bandwidth compared to a SNMP solution.*

1. Introduction

A Wireless Mesh Network (WMN) is a multi-hop wireless network with decentralized architecture, where each node can either communicate directly to other nodes or through intermediary nodes. These networks are highly tolerant to failures, being able to self-organize and self-configure themselves. In addition, WMNs have qualities such as ease of installation and maintenance, scalability, reliability and low cost. Therefore, they are widely used in community networks, disaster and emergency scenarios, home networks or extending corporate networks [Chung et al. 2013, Brito et al. 2014].

The Software Defined Networks (SDN) paradigm allows networks to become more intelligent and programmable. It separates control and data planes in network devices, proposing a centralized architecture to control packet forwarding. Such architecture greatly simplifies the network management since the control of the network is centralized. In Software Defined Wireless Mesh Networks (SDWMN), in-band control plane signaling is preferable over an out-of-band one since the former incurs on lower deploying costs by using the same network resources for control and data traffic [Brito et al. 2014].

In contrast, out-of-band control plane signaling requires a secondary network (virtual or physical) to transmit the control plane traffic, thus increasing network costs.

However, utilizing SDN in WMNs brings many challenges. Some of them are related to scalability and performance of the control plane, especially as the network grows in size and traffic. This way, multiple controllers become a natural strategy to approach both challenges of SDN control plane. Different approaches of multiple controllers are used to tackle scalability, performance and availability problems [Koponen et al. 2010, Tootoonchian and Ganjali 2010, Hassas Yeganeh and Ganjali 2012, Bari et al. 2013, Berde et al. 2014, Mattos et al. 2015], but they do not focus on wireless networks. Once WMN are sensible to quality and performance variations, these networks demand a resilient control plane and show strict requirements when compared to wired networks, specially in logically centralized networks. For example, these works do not take into account wireless in-band control traffic impact over the data plane traffic, as well as characteristics inherent to wireless environment, such as interference and others. Thus, it is desirable to have SDN controllers developed specifically for WMN when managing these networks.

Additionally, network monitoring is important to keep global network view consistent and to choose when and how to distribute the control plane. Monitoring metrics can be used for many network tasks, such as statistics plotting, events and alerts generation, traffic pattern analysis and as input for control plane applications and traffic engineering algorithms. Thus, most SDN controllers should require control and data planes monitoring system. Considering SDN controllers for wireless networks, the metrics gathered should be related to those networks. Standard SNMP and OpenFlow are not well suited for specific wireless data collection according to [Duarte et al. 2007, Nascimento et al. 2014], so custom agents are often required. From the best of our knowledge there are works, such as [Dely et al. 2011, Nascimento et al. 2014, Kim et al. 2016], closely related to, although they do not fully wrap together distributed controllers, wireless and monitoring.

Monitoring requires information sharing that generates traffic and processing costs which can be too high if not well balanced between precision and resource costs. Coupled with the dynamic quality of WMN links, the monitoring should ideally increase the monitored wireless-specific metrics while reducing the impact of monitoring costs over the data plane in an SDWMN in-band. In addition, the monitoring costs should follow the quality changes of wireless links.

To overcome these limitations, we propose in this paper a multi controllers architecture with adaptive monitoring in SDWMN in-band. Our approach allows a hierarchical distribution of the control plane with: i) a top level global controller (GC), which provides automated coordination configuration and management and maintains consistent global network view; ii) several local controllers (LC), where each one controls a set of switches, reducing control plane delay by reducing the distance from switches to controllers, and these LCs are deployed on-demand by GCs depending on network dynamics. Thus, this multi controllers architecture provides better scalability, performance and availability; iii) a passive monitoring approach where switches periodically send monitoring metrics to their LC without polling requests, reducing traffic costs; iv) adaptive monitoring where LCs can define in what interval and which metrics will be sent by switches according to

network conditions; v) switches have an agent which reads and captures specific wireless metrics not available in common SNMP or OpenFlow implementations.

This paper is divided as follows. Section 2 shows the background study. In section 3 we present the design and implementation of our architecture. The section 4 presents the evaluation and results. The paper concludes on section 5.

2. Background

Physically distributed control plane with logically centralized control plane requires state distribution between controllers. However, it can impact the performance of logically centralized control applications, this way, strongly consistent distributed controllers always operate with a consistent global view and thus, ensure that controllers operate properly in a coordinated manner. This process imposes overhead and delay, which can be especially harmful in SDWMN in-band, therefore limiting the responsiveness and may lead to sub-optimal decisions. Eventually consistent distributed controllers integrate the information as they become available, thus, the controllers react faster and can handle higher update rates, but tend to have a temporarily inconsistent global vision and possibly leading to incorrect behaviors. This results in two trade-offs: the first is between the overhead to ensure the consistency of the state distribution and performance of control applications; the second relationship is between the complexity of the logic of control applications and robustness to handle inconsistencies [Levin et al. 2012].

A single OpenFlow controller brings scalability problems to networks. To tackle these problems, several research efforts have been made. Works such as [Koponen et al. 2010, Tootoonchian and Ganjali 2010, Berde et al. 2014] propose a physical control plane distribution while maintaining the logical centralization using a distributed file system. In spite of distributing the control plane, these approaches impose a global network view in all controllers, consequently increasing the amount of shared information in the network, which is not desired in SDWMN in-band. Kandoo [Hassas Yeganeh and Ganjali 2012] proposes a distributed control plane that contains two-level hierarchical controllers, local applications are controlled by local controllers and the latter redirect decisions that need the global network view to a root controller. All communications between controllers are asynchronous and event-based as a publish subscriber architecture. In this method, a local controller works as a proxy for the requests coming from switches which would need to go directly to the root controller, resembling and providing the benefits of a proxy. This approach helps reducing the amount of messages exchanged over the control plane, which is desired in WMN. Other works deal with the amount and placement of controllers [Heller et al. 2012, Bari et al. 2013, Mattos et al. 2015].

These multi controllers approaches are generic and do not solve specific SDWMN problems. For example in an in-band scenario, the control plane communication uses the same channel as the data plane and it could negatively impact network traffic, therefore, being able to keep the overhead of control plane as low as possible is important. Also, to improve multi controllers support in SDWMN, the network should be able to deal with: link quality variations and interruptions, which may physically isolate controller instances from network devices; wireless forwarding devices can be many hops away from the controller, which would increase the latency and network convergence time; inconsistency

in network global view leads to performance degradation and errors in applications that need consistent state.

In terms of monitoring, these works have monitoring modules, which do not go deep into the subject. They collect few metrics such as delay or OpenFlow statistics and most of these metrics are actively measured (with periodic heartbeat or request messages inserted into network), which mostly are not specific for wireless networks management, such as RSSI or neighbor bytes and packets sent/received. Ideally, the monitoring should gather as many metrics as possible, thus enabling even more possible uses for them, while reducing the impact of monitoring costs over the data plane in an SDWMN in-band. Furthermore, link quality variations are often present in WMN due to many reasons (one of them, variable intra and inter-nodes interference), so a static monitoring overhead may degrade even further poor WMN links. Consequently, an adaptive monitoring overhead is desired [Hava et al. 2012]. [Kim et al. 2016, Shah et al. 2016] recently addressed monitoring in ONOS framework [Berde et al. 2014], which has multi controllers feature. The first records OpenFlow message exchanged on the ONOS logging system, providing real time monitoring result. The latter is an adaptive monitoring solution to dynamically monitor the overall load of individual controllers working in a logically centralized SDN environment. However, both of them do not focus on monitoring wireless-specific metrics, neither on network performance metrics.

Common protocols used for network monitoring in SDWMN are SNMP and OpenFlow. However, they are not well suited for specific data collection from wireless physical and link layers [Duarte et al. 2007, Nascimento et al. 2014] and it directly impacts the flexibility benefits that SDN could have in WMN. [Nascimento et al. 2014] implements extensions to OpenFlow to contemplate specific wireless functionalities, including new messages into OpenFlow to query nodes for wireless interfaces physical layer metrics, but it was only tested in an out-of-band scenario. [Dely et al. 2011] and [Rethfeldt et al. 2015] insert custom SNMP agents in WMN client and uses non OpenFlow router nodes, respectively, to collect specific wireless metrics from the WMN. Although, the first is too end-user intrusive and the latter has a relative high monitoring traffic cost of 80Kb for each mesh router per monitoring cycle.

Works focused on reducing network resource utilization in WMN or on adaptive monitoring overhead according to network quality are not based on SDN. A hierarchical WMN monitoring is presented in [Nanda and Kotz 2008] where a node is responsible for monitoring itself and nodes k-hopes distant. The monitoring data gathering is passive, decentralized and uses a combination of passive and active monitoring techniques depending if the amount of monitoring overhead in a link is high or low, respectively.

Related works show that there are few approaches aggregating granular metrics from physical and link layers of SDWMN while reducing network resource utilization in these networks. The latter is important because it can negatively impact network performance in a SDWMN in-band [Hava et al. 2012].

3. General Architecture

Our work proposes a multi controllers architecture with adaptive monitoring in SDWMN in-band. Figure 1 shows an example of this architecture with a global controller (GC), a monitoring database (MonitDB) and *N* control domains. Each control domain consists of

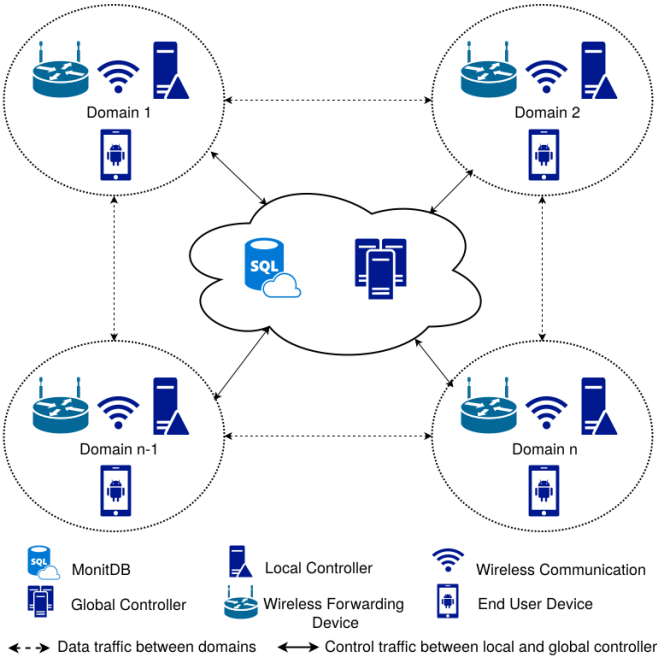


Figure 1. System Architecture

a Local Controller (LC) and wireless forwarding devices (WFD) with OpenFlow support, such as wireless mesh router (WMR) and access points (AP). Lastly, the end user devices connect to one of the forwarding devices inside a control domain, as if the end user was connected to a common access point. This architecture was implemented as a proof-of-concept on OpenWiMesh [Brito et al. 2014], a framework for developing SDN in WMN in-band, extending its features.

Hierarchical Architecture: To enable faster answers to WFD requests and to reduce control plane overhead in the rest of the network, our architecture is based on a hierarchy of GC and LCs. This approach offloads control plane applications load over the multiple controllers, which is more suitable for a WMN unlike the horizontal approach that impose the network-wide state synchronization cost.

Global Controller: One top level global controller provides automated coordination, configuration and management of local controllers and maintains consistent global network view. The GC provides top-level applications to LCs through an Global Control API (GCAPI). GCs can reactively respond to LC's GCAPI queries, e.g inter-domain routes discovery, or proactively distribute commands to LCs, e.g coordinate the deploying and installation of OpenFlow rules in LCs and delegation of data plane devices to LCs. The GC subscribes to each LC to receive periodical topology information updates from them, so the global network view is updated in an event-based communication, as later seen in Figure 2. We assume that the GC is deployed in a cloud cluster with high availability, for example, in one or more data centers.

Local Controllers: LCs are dynamically provisioned and deployed on-demand by the GC when and where needed according to the network dynamics possibly based on data plane overhead, network fault tolerance or simply by network policies. Each LC controls a set of WFD through OpenFlow (thus creating a control domain) and manages

the entire intra-domain information. LCs do not communicate with each other and they only know about nodes inside their domain. This way, the GC must be queried by LCs, for example, about packet forwarding to nodes outside LC's domain. In our implementation, the LC is represented by the controller element that is inside the WMR as shown in [Brito et al. 2014] and our work extends the WMR, making it remotely accessible and having its controller element started/stopped by the GC. Although, a LC could also be deployed in a similar way by the Network Function Virtualization (NFV) concept in a server close to domain devices. Moreover, every WMR is a potential controller, so the number of LC can follow the network growth.

Global Control API: LCs can access the applications provided by GC using a remote access API. The GCAPI is implemented using a Python library called Pyro[1]. Each LC has a GCAPI client installed that calls the methods implemented in GC.

Inter Domain Routing Table: The Inter Domain Routing Table (IDRT) is present in LCs and provides instructions required to forward packets to outside domain destinations. In a few words, entries in IDRT are made of different metrics and each entry correlates two domains, for example there must be a field for the local domain edge node and another for neighbor domain edge node, if additional metrics are required they may be requested to the MonitDB. This table can optionally lower the amount of control traffic over the network by preventing LCs from constantly requesting outside domain information to the GC in a short spam of time, by working as a cache. The IDRT implemented in this work is adapted from [Phan et al. 2013].

Inter Domain Routing: If a new flow arrives at a WFD, and if this flow does not match any flow entry, the WFD forwards a PacketIn message to LC that controls him. If the destination of this flow is not within the local domain, the LC looks up in the IDRT for instructions on how to forward this flow to another domain. Once the destination is not in IDRT, the LC sends this request to GC by using the GCAPI. Then, the GC performs path computations and send the routing instructions to all LCs which control the domains where the flow request will be forwarded to. So each LC parses GC's instructions and install the forwarding rules of this flow into their local domain WFDs.

Fault Tolerance: We consider 4 types of failures: path, LC and GC failures and GC isolation. In the first one, OpenWiMesh framework can find new routes once network links of a path fail. Secondly, if a LC fails, the nodes of its domain must reconnect to another LC. So, local domain edge nodes of the failed LC will connect to the LC of the closest neighbor domain and the local nodes near the edge nodes will successively follow the edge nodes by connecting on same LC. The GC is responsible to define whether the failed domain nodes will be integrated into the new domain or if a new controller will be instantiated to manage these nodes. Thirdly, as described before, we assume that the GC is deployed in a high availability (HA) environment. Finally, in case of a domain isolation, when a LC loses communication with GC and there is no neighbor domain to forward GC's traffic, the LC of this isolated domain keeps accepting new flows for local traffic only. Meanwhile, the LC keeps trying to communicate with the GC through every local domain edge node and, once it succeeds, the communication with GC is reestablished.

Monitoring Database: As shown in Figure 2, the Monitoring Database

[1]https://pythonhosted.org/Pyro4/

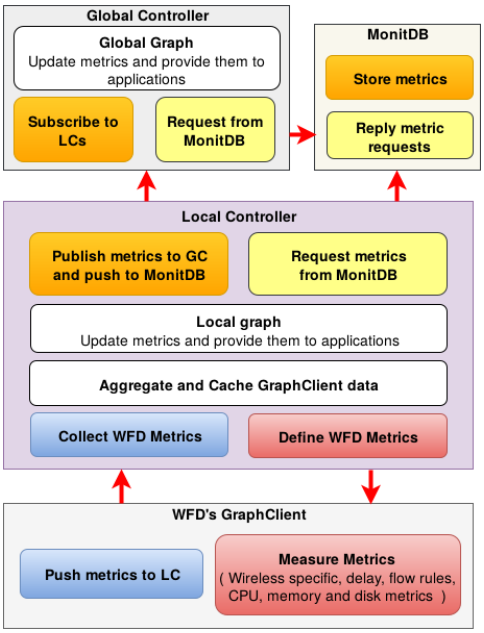


Figure 2. Proposed monitoring system

(MonitDB) can be consulted by GCs and LCs and have data inserted by LCs with monitoring data metrics from their local network view. Apart from sending simple local domain network topology data to GC, LCs periodically write network metrics from its domain to the MonitDB. The periodic push mode from LC refers to the LC working as a monitoring proxy, temporarily aggregating monitoring data before transmitting it to the MonitDB, this way no data is lost in case of communication problems between them. LCs can send their raw or calculated network metrics, such as bytes transferred or average local domain intra-node delay. The Monitoring Database can be accessed by LCs or GCs when querying for historical data metrics or inter domain routing metrics, respectively.

Network Graph: Data structure responsible for storing local network view data in the LCs (local network graph) and global network view in GC (Global network graph), as shown in Figure 2. LCs update their local view with the information periodically sent by the GraphClient module running in every local domain WMR, as described in [Brito et al. 2014]. LCs also periodically send their local view to the GC, but any change in network topology of any domain is immediately reported to the GC. GC has the entire network topology view, but specific domain metrics are consulted on the MonitDB, which are periodically populated by LCs.

Proactive Push-Based Monitoring: In a WMN, a single fault can affect several neighboring nodes in a network, and even disconnecting them, leading to isolated partitions. These faults could make the gathering of accurate network information challenging or even impossible after the occurrence of the fault. Hence we advocate the use of a periodic and proactive approach to monitor the network in order to be able to detect fault conditions before it actually happens. Aiming for monitoring traffic overhead reduction, a push-based mechanism is used, where nodes proactively send monitoring information to the LC without receiving queries messages from it.

GraphClient: GraphClient is an agent module installed in every WFD in the

network to passively collect specific metrics from wireless interfaces. The wireless monitoring metrics are collected via the Linux firmware nl80211 and sent to the LC using a periodic proactive push-based approach. GraphClient's role is not to distribute monitoring information to other WFDs, but to keep the LC graph up-to-date with network quality variations (as seen in Figure 2) and to enable a historical view on network quality. Our paper extends GraphClient functionalities [Brito et al. 2014] by inserting automatic/dynamic variation of data pushing intervals and number of metrics monitored (based on LC's control), as well as new monitoring metrics as shown in Table 1[2]. The new metrics enable better network traffic and load behavior view as well as providing ground for new different management applications and routing algorithms. With these metrics one can, for example, detect overloaded WFD nodes or links (as done by our adaptive monitoring) or create paths over low latency links or low load nodes.

Type of data	Description
Node metrics	CPU usage* (%)
	TX Power (dB)
	Free Memory (MB)*
	Free disk space (MB)*
	Openflow flow rules statistics*
Association metrics	RSSI
	Bytes sent*
	Bytes received*
	Packets sent*
	Packets received*
	Packets retransmitted
	Max bandwidth (Mbps)
	Inactive time (s)
	Bidirectional delay (s)*

Table 1. Metrics captured by GraphClient.

Adaptive Monitoring: The monitoring overhead inside a local domain follows the network quality of the own local domain. Each LC monitors the network quality inside its domain and, based on the quality of each WFD and network wireless links, it controls in what interval and which variables each GraphClient will send its monitoring metrics back to the LCs. If one of the network wireless links drops in quality and there is no other better route for the monitoring traffic, then the LC tell all WFDs reachable through that link to increase the interval of monitoring or, at the same time, lower the number of metrics monitored. Once the link quality gets back to normal, LC can tell the node to monitor again all variables back at the lowest interval. One thing to keep in mind is to not over increase the measurement interval of metrics used by routing or network quality detection algorithms in LCs, thus avoiding long network graph inconsistencies. Its encouraged to keep these routing metrics always monitored in fixed intervals, while others can be changed or not even monitored.

[2]In Table 1, the fields with '*' are the new metrics of our GraphClient extension.

In our implementation, LCs can set 3 different monitoring states, *normal(N)*, *higher interval(HI)* and *minimum(M)*. At *N* state, all monitoring variables are measured and collected at the lowest pre-defined interval. If the bandwidth consumption of a link goes past a threshold (25%) of the residual bandwidth, LCs order WFDs to change to the *HI* state. In *HI* state, all monitoring variables have their collection interval set *Y* times higher, except the ones that are being used for the routing algorithms, otherwise the network graphs gets less updated and the network may become less stable. If the bandwidth consumption of a link is over 50%, only the variables used for routing are kept being monitoring in the normal interval while all others are not measured or collected anymore. The thresholds selected are not based on researches, to the best of our knowledge, due to the lack of it. Nevertheless, the chosen thresholds aims to lower the monitoring traffic overhead, consequently lowering the possibility of losses and physical modulation changes in a saturated wireless link before it happens. All states can have intervals and amount of metrics configurable.

4. Evaluation and Results

We evaluated our proposal in two stages. In the first stage we execute three experiments, the first two of them to demonstrate the scalability improvement, based on [Hassas Yeganeh and Ganjali 2012] evaluations, and a third to show the lower switch-controller latency brought by our architecture. In all three experiments we compare the results to a single OpenWiMesh controller (Normal OWM). In the second stage, we show that our monitoring solution captures different metrics including wireless ones as expected and then, we compare the amount of monitoring bandwidth a node would expend using either SNMP or our monitoring solution to show how much our solution saves network bandwidth compared to common solutions. These evaluations demonstrate the feasibility of multi controllers and monitoring of specific wireless metrics of a SDWMN in-band, in order to distribute the control plane in an WMN and provide scalability and enhanced control plane management. The proposed architecture has a lot of features, for example, fault tolerance and adaptive monitoring. But, for the sake of space, we are focusing on testing scalability, latency and monitoring metrics.

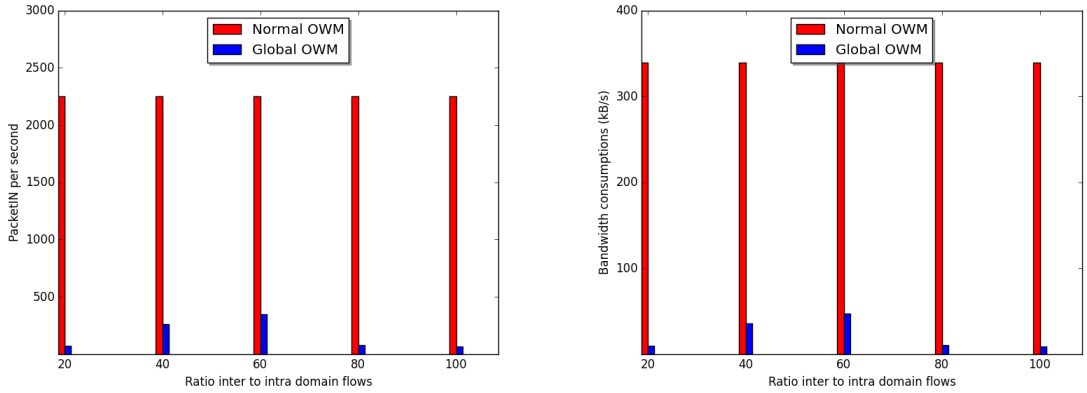
The adopted topology for the experiments is equivalent to the depicted in Figure 1, with a fan out factor varying from 2 to 4 (i.e. 2 to 4 local controllers and 10 WFD per local domain). Additionally, nodes dispersion were randomly done by CORE Emulator and we considered the following assumptions: Every node is at most 5 hops away from its controller and every node has 10 ms delay from each other configured from Linux traffic control *tc* command.

We evaluate our proposal in an emulated environment using the same real time emulator described in [Brito et al. 2014] [3]. This emulator is capable of emulating devices operating in data and network layers, including the physical layer operating properties of them. At its emulation scenario in GNU/LINUX, it utilizes kernel namespaces to create devices and Linux network bridges to create link connections between nodes. Each node has its own processes and environment variables, thus creating a totally isolated environment between network nodes. To enable the collection of neighbors bytes and packets (as shown in Table 1), we had to setup the emulator to, instead of creating namespaces with

[3]CORE Emulator v4.8

its default network wireless interfaces, create network interfaces using the kernel module called mac80211_hwsim. We plan to implement link losses and medium access delay in our future tests. To run the experiments, we used a Personal Computer with an Intel(R) Core(TM) i7-3630QM 2.40GHz CPU and 8 GB of RAM, running Debian 9. We executed every test 10 times and we only show here the averages taken with 95% confidence interval.

In the first two experiments of the first stage, we measure the number of requests (PacketINs) processed by each controller and their bandwidth consumption. Our main goal is to decrease the load on our global OpenWiMesh controller (Global OWM) compared to the Normal OWM. We evaluate how the control plane scales against the ratio of inter and intra domain flows (variating inter domain traffic from 20% to 100%). Each WFD starts fifty TCP flows to any other WFD on the network. Figure 3 shows that the Global OWM load is much lower than the Normal OWM, even when 100% of traffic are inter domain. This happens because the IDRT presents on LCs work as a cache reducing the amount of requests to the GC, as soon as the inter domain traffic grows and more "cache hits" happen. In the second experiment, the ratio of inter and intra domain flow is set to 20% using different fanout (from 2 to 4). The goal of this experiment is to measure our proposal's control plane scalability as the number of nodes increases. As seen in Table 2, the larger the network is, the larger the load difference is between Normal OWM and our proposal. It happens because a local controller works as a proxy for the requests coming from WFDs protecting the GC from high request load. Meaning that our proposal scales better than a single OpenFlow controller, such as the Normal OWM.



(a) Average number of PacketIns received by the controllers

(b) Average number of bytes received by the controllers

Figure 3. Control Plane load for inter and intra domain routing flows scenario. The load is based on the number of inter domain flows.

In the third experiment of stage one, we measure the average latency between WFD and their respective controllers via ICMP ping's RTT. Each WFD is in ping loops for 10 minutes, sending 3 ping packets in 1 second interval each to WFDs, in a crescent IP order. There is a 10 ms delay configured on each hop and ARP resolutions delay are accounted. The goal here is to show the positive impact in switch-controller latency as more LCs are deployed in the network. Table 3 presents the average switch-controller latency varying the fan out factor, as seen in the second experiment. The result shows lower latencies for our architecture in every fan out factor. It happens because each LC

Number of nodes	Normal OWM (kB / s)	Global OWM (kB / s)	Normal OWM (pktINs / s)	Global OWM (pktINs / s)
Fan out 2	156	8.8	1020	60
Fan out 3	238	9.6	1625	65
Fan out 4	339	11.3	2250	73

Table 2. Control Plane load for inter and intra domain routing flows scenario. The load is based on the number of nodes.

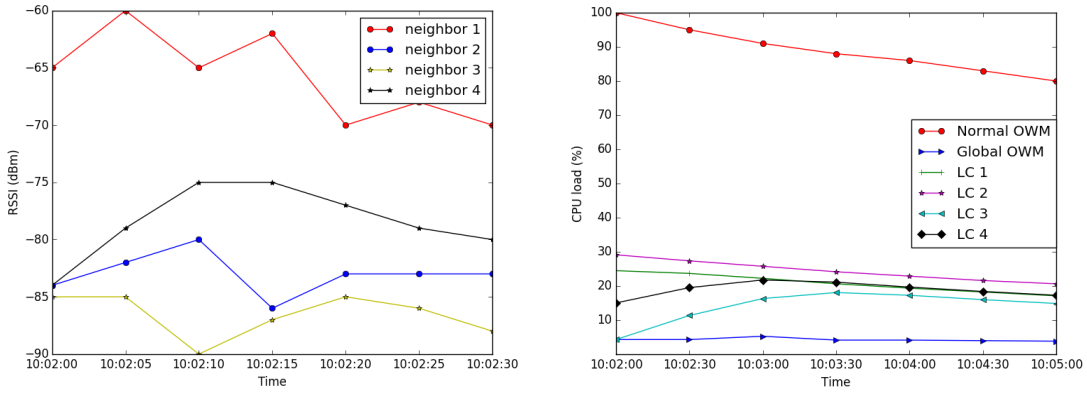
controls a set of switches, reducing control plane delay by reducing the distance from switches to controllers compared to a single controller topology.

Number of nodes	Normal OWM	Multiple Controllers
Fan out 2	23 ms	15 ms
Fan out 3	29 ms	16 ms
Fan out 4	34 ms	18 ms

Table 3. Differences in average controller-switch latency with a single and multiple controllers.

The experiments of the second stage are done in a fan out 4 topology (40 nodes) with 20% inter domain flows ratio. In the first experiment of the second stage, we show the time series monitoring metrics captured by our solution, including the wireless-specific metrics and the CPU use of normal controllers and our solution's. The goal here is to show that our solution scales linearly and enables better network view with wireless-specific metrics. For the sake of space, we show here some of the metrics captured. Figure 4 shows two time series with metrics captured from GraphClient. In Figure 4(a) we show a time series with the RSSI values related to each neighbor of one WFD in the network, proving us with all link connection qualities in the network and helping, for example, the decision of routing algorithms. This kind of time series can be done with all metrics monitored, providing historical and better network view as a whole. Figure 4(b) shows the CPU load behavior of a single Normal OWM and our proposal's controllers. The average CPU load in the global and local controllers of our proposal is lower than the load on the single OWM controller, demonstrating our load distribution and scale potential. CPU load metrics can help us deciding when and where to deploy these multiple controllers.

In the second experiment, we compare the average monitoring bandwidth costs of a WFD trying to deliver all its possible monitoring metrics running either our proposed monitoring system or a common implementation of SNMP. We also change the number of wireless neighbors of a WFD from 1 to 5, basically repeating the metrics for each neighbor and rising the packet size. The goal is to show how much our proposal's monitoring system can save bandwidth compared to a common monitoring protocol. The WFD's metrics monitored are the ones shown in Table 1, except the "Openflow flow rule statistics" field, because it varies too much depending on the number of OpenFlow rules installed in the WFD. Our calculations in this experiment are based in the protocols overheads shown in Figure 5 and we consider each metric as a SNMP OID where each OID has a fixed fair size of 10bytes. As seen in Table 4, our solution saves, at minimum, near 50% in



(a) RSSI time series of neighbors of a WFD

(b) CPU load time series on a Normal OWM and our solution's controllers

Figure 4. Time series graphs with different monitoring metrics captured by GraphClient.

bandwidth compared to a SNMP reply, because there's a considerable OID overhead in SNMP which is minimum in our solution. Besides, in periodical monitoring using SNMP protocol, there is also the request messages overhead in the network, while there are none periodic requests in our solution.

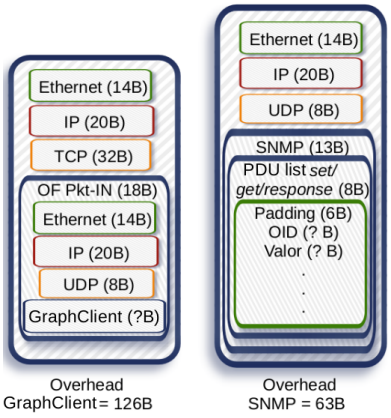


Figure 5. Protocols headers overhead on monitoring packets of our monitoring system's and SNMP's packets

5. Conclusion and Future Work

SDWMN in-band can join the benefits from SDN and WMN to make WMN more intelligent and programmable without high deployment costs. However, there are some issues about control plane scalability, performance, and availability. To address these issues several works in the literature proposed multiple controllers architectures, but these proposals were not implemented to SDWMN in-band. The implementation of a multi controllers architecture in SDWMN in-band requires to lead with challenges associated with the wireless mesh environment. Besides, a monitoring system able to gather specific wireless metrics without overloading the communication channel is indispensable.

We proposed a multi controllers architecture with adaptive monitoring in SDWMN in-band which objective is to reduce control plane overhead and delay while im-

Bandwidth cost (bytes)	**1 Neighbor**	**2 Neighbors**	**3 Neighbors**	**4 Neighbors**	**5 Neighbors**
SNMP	387	615	854	1084	1274
Our solution	211	286	355	415	497

Table 4. Comparison of different monitoring packet size averages

proving network scalability. The results show that our architecture linearly scales the control plane while reduces switch-controller latency, as well as offering better network view and management for wireless networks while providing low monitoring overhead compared to common monitoring protocols. To the best of our knowledge, this is the first proposal of distributed controllers for SDWMN in-band with wireless-specific metrics and adaptive monitoring.

As future work, we plan to evaluate the elasticity of our architecture with several dynamic controller placement algorithms and evaluate the benefits of adaptive monitoring in lowering packet drops in the network. We also plan to implement link losses and medium access delay in our tests.

References

Bari, M., Roy, A., Chowdhury, S., Zhang, Q., Zhani, M., Ahmed, R., and Boutaba, R. (2013). Dynamic controller provisioning in software defined networks. In *Network and Service Management (CNSM), 2013 9th International Conference on*, pages 18–25.

Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., Lantz, B., O'Connor, B., Radoslavov, P., Snow, W., and Parulkar, G. (2014). Onos: Towards an open, distributed sdn os. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, pages 1–6, New York, NY, USA. ACM.

Brito, I., Gramacho, S., Ferreira, I., Nazare, M., Sampaio, L., and Figueiredo, G. (2014). Openwimesh: A framework for software defined wireless mesh networks. In *Computer Networks and Distributed Systems (SBRC), 2014 Brazilian Symposium on*, pages 199–206.

Chung, J., Gonzalez, G., Armuelles, I., Robles, T., Alcarria, R., and Morales, A. (2013). Experiences and challenges in deploying openflow over real wireless mesh networks. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, 11:955–961.

Dely, P., Kassler, A., and Bayer, N. (2011). Openflow for wireless mesh networks. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–6. IEEE.

Duarte, J. L., Passos, D., Valle, R. L., Oliveira, E., Muchaluat-Saade, D., and Albuquerque, C. V. (2007). Management issues on wireless mesh networks. In *2007 Latin American Network Operations and Management Symposium*, pages 8–19.

Hassas Yeganeh, S. and Ganjali, Y. (2012). Kandoo: A framework for efficient and scalable offloading of control applications. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 19–24, New York, NY, USA. ACM.

Hava, A., Ghamri-Doudane, Y., and Murphy, J. (2012). On monitoring overhead impact in wireless mesh networks. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International*, pages 487–492. IEEE.

Heller, B., Sherwood, R., and McKeown, N. (2012). The controller placement problem. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 7–12, New York, NY, USA. ACM.

Kim, W., Li, J., Hong, J. W. K., and Suh, Y. J. (2016). Ofmon: Openflow monitoring system in onos controllers. In *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pages 397–402.

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., and Shenker, S. (2010). Onix: A distributed control platform for large-scale production networks. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, pages 1–6, Berkeley, CA, USA. USENIX Association.

Levin, D., Wundsam, A., Heller, B., Handigol, N., and Feldmann, A. (2012). Logically centralized?: State distribution trade-offs in software defined networks. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 1–6, New York, NY, USA. ACM.

Mattos, D. M. F., Lopez, M. A., Ferraz, L. H. G., and Duarte, C. M. B. (2015). Controlador resiliente com distribuição eficiente para redes definidas por software. In *Computer Networks and Distributed Systems (SBRC), 2015 Brazilian Symposium on*.

Nanda, S. and Kotz, D. (2008). Mesh-mon: A multi-radio mesh monitoring and management system. *Comput. Commun.*, 31(8):1588–1601.

Nascimento, V., Moraes, M., Gomes, R., Pinheiro, B., Abelém, A. J. G., Borges, V. C. M., Cardoso, K. V., and Cerqueira, E. (2014). Filling the gap between software defined networking and wireless mesh networks. In Raz, D., Nogueira, M., Madeira, E. R. M., Jennings, B., Granville, L. Z., and Gaspary, L. P., editors, *CNSM*, pages 451–454. IEEE Computer Society.

Phan, X. T., Thoai, N., and Kuonen, P. (2013). A collaborative model for routing in multi-domains openflow networks. In *Computing, Management and Telecommunications (ComManTel), 2013 International Conference on*, pages 278–283.

Rethfeldt, M., Danielis, P., Moritz, G., Konieczek, B., and Timmermann, D. (2015). Design and development of a management solution for wireless mesh networks based on ieee 802.11s. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 902–905.

Shah, S. A. R., Bae, S., Jaikar, A., and Noh, S.-Y. (2016). An adaptive load monitoring solution for logically centralized sdn controller. In *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 1–6.

Tootoonchian, A. and Ganjali, Y. (2010). Hyperflow: A distributed control plane for openflow. In *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, INM/WREN'10, pages 3–3, Berkeley, CA, USA. USENIX Association.

Trilha Principal do SBRC 2017
Sessão Técnica 16
Redes de Sensores e Redes Corporais

Escalonamento de Nós em Redes Aquáticas Estratificadas utilizando Voronoi

Eduardo P. M. C. Júnior[1], **Luiz F. M. Vieira**[1], **Marcos A. M. Vieira**[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)
31270-901 – Belo Horizonte – MG – Brasil

{epmcj,lfvieira,mmvieira}@dcc.ufmg.br

Abstract. *Underwater networks are usually used for monitoring water resources and underwater environments. Thus, it is important that the underwater sensor nodes cover the largest region possible, during the largest amount of time. This paper presents a method to perform node scheduling in underwater stratified networks. It aims to maintain the network active for longer, maintaining its connectivity. Voronoi Diagrams are used to decompose the space into regions around each node in order to determine which one should be scheduled to sleep. Simulation results show that the proposed method achieves the desired objectives, more than doubling the network lifetime while guaranteeing connectivity.*

Resumo. *Redes aquáticas são comumente utilizadas para fins de monitoramento de recursos hídricos e de ambientes aquáticos. Logo, é importante que seus nós cubram a maior região possível durante a maior quantidade de tempo. Este artigo apresenta um mecanismo para realizar o escalonamento de nós em redes aquáticas estratificadas e que visa mantê-las ativas por mais tempo. Diagramas de Voronoi são utilizados para decompor o espaço em regiões que cercam cada um dos nós e então decidir se é possível desativar alguns deles temporariamente. Resultados das simulações mostram que o método proposto alcança os objetivos almejados, mais que dobrando o tempo de vida da rede enquanto garante sua conectividade.*

1. Introdução

Estudos e monitoramentos de ambientes aquáticos são de grande relevância para as atividades humanas. Como tais ambientes causam grande impacto em nossas vidas, seja através da água que bebemos ou das mudanças climáticas pelos quais são responsáveis, é necessário se ter um bom conhecimento sobre eles. Nesse sentido, a coleta de dados desses ambientes é essencial e pode ser realizada utilizando redes aquáticas.

Essas redes são responsáveis por realizar a comunicação entre seus elementos de modo a permitir que uma cobertura adequada do ambiente seja alcançada. Diferentemente das redes terrestres, os equipamentos utilizados em redes aquáticas geralmente apresentam preços muito elevados e o gerenciamento de energia é algo ainda mais crítico, uma vez que o fornecimento pode ser impraticável. Assim, é necessário que os nós da rede realizem a cobertura da região desejada durante o maior tempo possível [Vieira et al. 2010].

Neste artigo, é proposto um mecanismo[1] para realizar o escalonamento de nós em

[1]Código disponível em `https://github.com/epmcj/uwnodescheduling`.

redes aquáticas estratificadas, onde nós sensores e dispositivos são posicionados através de diversas camadas, cada uma em uma profundidade diferente. Ele visa aumentar o tempo de vida da rede, mantendo sua conectividade. São utilizados diagramas de Voronoi para determinar a importância de cada um dos nós e selecionar quais deles podem ser desativados temporariamente.

A avaliação do método é feita através de simulações. Os resultados mostram que a solução proposta consegue atingir seus objetivos e indicam um aumento do tempo de vida da rede quando ele é utilizado.

As principais contribuições deste artigo são: descrição do método proposto para escalonamento de nós baseado no diagrama de Voronoi, avaliação experimental com vários parâmetros e resultados que mostram que o tempo de vida da rede é aumentado.

O artigo está organizado da seguinte forma. A próxima seção discute os trabalhos relacionados. A seção 3 detalha os conceitos preliminares. A seção 4 é utilizada para descrever o método proposto. A seção 5 traz os resultados obtidos e as análises deles. Finalmente, a seção 6 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Devido à crescente atenção dadas às redes aquáticas, muitos trabalhos tem sido desenvolvidos para resolver problemas encontrados nelas. Alguns protocolos foram desenvolvidos para a camada de enlace, como o Aloha [Vieira et al. 2006]. Outros protocolos foram feitos para a camada de rede, como o Pressure Routing [Lee et al. 2010], GEDAR [Coutinho et al. 2014], usando roteamento oportunístico [Vieira 2012, Coutinho et al. 2016d], roteamento geográfico [Coutinho et al. 2016a], com controle de profundidade [Coutinho et al. 2013] ou baseados em centralidade [Coutinho et al. 2016c]. Nenhum deles investiga o escalonamento de nós sensores aquáticos.

Técnicas para roteamento que consideram que os nós possam dormir por intervalos de tempo são apresentadas em [Coutinho et al. 2015]. O desenvolvimento de protocolos e algoritmos que possam economizar energia da rede são discutidos em [Coutinho et al. 2016b]. No entanto, nenhum deles propõe o uso de diagramas de Voronoi para escalonar os nós sensores, preservando a energia dos nós sensores e aumentando o tempo de vida da rede.

Em [Vieira et al. 2003], é desenvolvida uma solução para o escalonamento de nós sensores para redes de sensores terrestres que é baseada em diagramas de Voronoi. Com ela os autores conseguem poupar energia da rede sem diminuir sua área de sensoriamento. Não é apresentado um estudo para redes aquáticas ou redes 3D estratificadas, nem um estudo sobre o número de camadas.

Outros problemas em redes de sensores aquáticas também contam com soluções que utilizam diagramas de Voronoi. O problema de otimização de cobertura de redes aquáticas é um exemplo [Wang and Wang 2016]. Em [Wu et al. 2013], os autores propõem um mecanismo que utiliza diagramas de Voronoi para ajustar a profundidade dos nós de uma rede e assim maximizar sua cobertura. Vale notar que nenhum dos dois escalonam os nós sensores.

3. Conceitos Preliminares

Nessa seção são apresentados conceitos preliminares relevantes para este trabalho. Primeiramente, o modelo da rede considerada é descrito. Em seguida, tem-se uma breve descrição sobre diagramas de Voronoi, seguida pela exposição do modelo utilizado na estimativa de erros em entregas de pacotes em redes aquáticas.

3.1. Modelo da Rede

A cobertura de uma região aquática por nós de uma rede pode ser feita de diversas maneiras. Uma delas consiste em definir camadas em determinadas profundidades de interesse e então distribuir nós dentro de cada uma delas. Redes que utilizam essa forma são chamadas de estratificadas. Essas são as redes consideradas no desenvolvimento do método proposto. A Figura 1 mostra o exemplo de uma rede estratificada que possui 3 camadas.

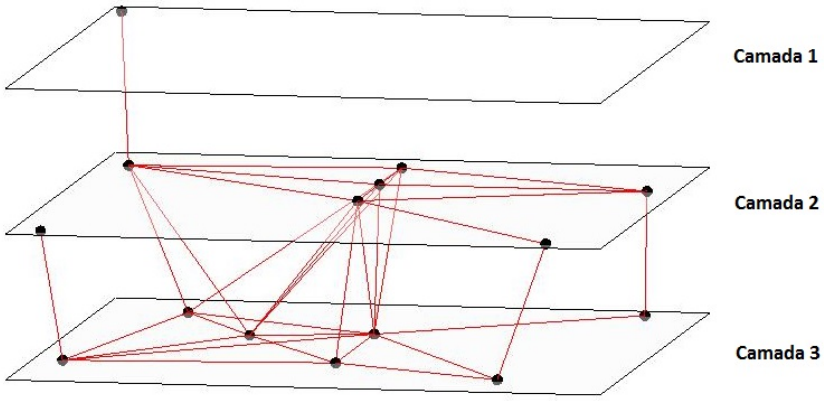


Figura 1. Exemplo de rede estratificada

É esperado que os dados da rede fluam no sentido de baixo pra cima, ou seja, que os nós das camadas mais profundas transmitam mensagens para aqueles que estão mais próximos da superfície. O protocolo de roteamento proposto em [Noh et al. 2016], por exemplo, utiliza a pressão dos nós da rede para rotear dados de nós submersos até as boias que se encontram na superfície.

Cada camada da rede é considerada como sendo um plano 2D, uma vez que todos os nós dentro dela possuem a mesma profundidade. É esperado que elas contenham pelo menos um nó e que cada uma consiga se comunicar com no mínimo mais uma, caso contrário a rede conteria um componente desconectado.

Em diferentes instantes de tempo pode ser feita a avaliação que considera a posição de cada um dos nós sensores. Essa posição pode ser global ou relativa a algum ponto. Mesmo que *Global Positioning Systems* (GPS) não funcionem muito bem em ambientes aquáticos [Akyildiz et al. 2005], pode-se utilizar abordagens que utilizem técnicas de localização [Erol et al. 2007b, Erol et al. 2007a, Erol et al. 2008].

A topologia da rede é considerada como sendo dinâmica. Isso se deve ao fato de que seus nós podem ser desativados temporariamente ou desligados definitivamente.

3.2. Diagrama de Voronoi

Seja $S = \{p_1, p_2, ..., p_n\}$ um conjunto de pontos em um plano Euclidiano. Esses pontos são chamados sítios. Seguindo a definição encontrada em [de Berg et al. 2008], tem-se que o Diagrama de Voronoi de S corresponde às n subdivisões do plano, chamadas de células, onde cada ponto q pertence à célula do sítio p_i se e somente se a distância entre q e p_i é menor do que a distância entre q e p_j, para todo $p_j \in S$ com $j \neq i$. Logo, cada sítio irá determinar uma célula cujo o tamanho dependerá da posição dos demais sítios. Assim, a célula $V(p_i)$ do sítio p_i pode ser expressada como:

$$V(p_i) = \{q : |p_i - q| \leq |p_j - q|, \forall j \neq i\} \quad (1)$$

A Figura 2 mostra os diagramas de Voronoi de um exemplo de uma rede estratificada com 3 planos. O plano do meio é destacado para exemplificar um diagrama de Voronoi.

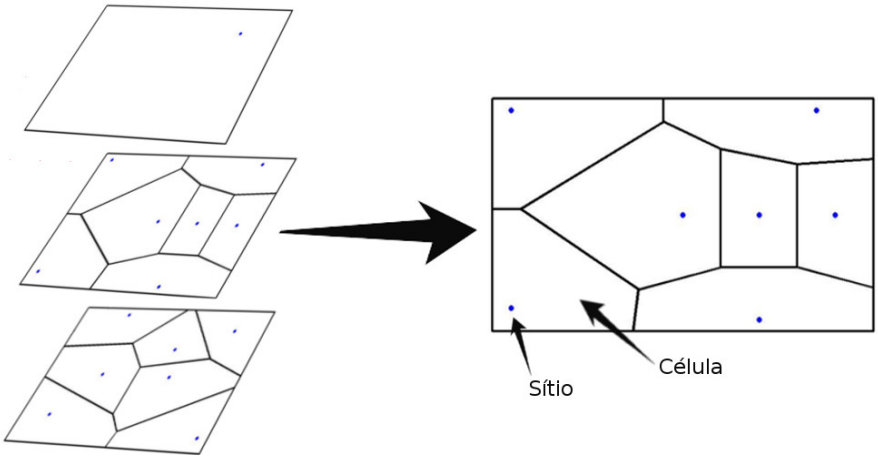


Figura 2. Diagrama de Voronoi dos pontos.

3.3. Estimativa da Probabilidade de Erros em Entrega de Pacotes

Nessa subseção é descrito um modelo de propagação de sinal em ambientes aquáticos para estimar a probabilidade de erros na entrega de pacotes. Ele é caracterizado por uma atenuação do sinal e por ruídos presentes no meio [Stojanovic 2007]. A atenuação, em dB, que ocorre em um canal acústico aquático sobre uma distância d, para um sinal de frequência f, é dado por

$$A(d, f) = 10k \times \log d + d \times \alpha(f) \times 1000 + 10 \quad (2)$$

Como dito em [Brekhovskikh 2003], o primeiro termo desta equação representa a perda por espalhamento, enquanto o segundo representa a perda por absorção. O fator de espalhamento k é utilizado para descrever a geometria da propagação. Seus valores comuns são $k = 2$, para o espalhamento esférico, $k = 1$, para espalhamento cilíndrico, e $k = 1.5$ para espalhamento prático. A perda por absorção pode ser determinada empiricamente pela fórmula de Thorp [Brekhovskikh 2003]. Para frequências maiores do que algumas centenas de Hz, essa fórmula é:

$$10 \log \alpha(f) = 0.11 \frac{f^2}{1+f^2} + 44 \frac{f^2}{4100+f^2} + 2.75 \times 10^{-4} f^2 + 0.003, \quad (3)$$

onde $\alpha(f)$ é dado em dB/km se f é dado em kHz.

Os ruídos podem ser modelados através de quatro fontes: turbulência (N_t), atividade de embarcações (N_s), ondas (N_w) e ruídos térmicos (N_{th}) [Coates 1989]. Assim, o ruído total ($N(f)$) do ambiente é dado pela soma dos valores dessas fontes. As seguintes fórmulas empíricas mostram uma maneira de calcular os valores dos ruídos de cada uma das quatro fontes, em dB re μ Pa per Hz, em função da frequência f em kHz:

$$\begin{aligned} 10 \log N_t(f) &= 17 - 30 \log f \\ 10 \log N_s(f) &= 40 + 20(s - 0.5) + 26 \log f - 60 \log(f + 0.03) \\ 10 \log N_w(f) &= 50 + 7.5 w^{\frac{1}{2}} + 20 \log f - 40 \log(f + 0.4) \\ 10 \log N_{th}(f) &= -15 + 20 \log f \end{aligned} \quad (4)$$

onde s é o fator de atividade de embarcações, com valor entre 0 (pouca atividade) e 1 (muita atividade), e w é a velocidade do vento, em m/s.

Com os valores da atenuação $A(d, f)$ e do ruído $N(f)$ é possível calcular o valor da relação sinal-ruído (SNR, do inglês *Signal-to-Noise Ratio*) observado sobre uma distância d, com frequência de transmissão f e utilizando uma potência P [Stojanovic 2007]. Sendo Δf a largura de banda de ruído do dispositivo receptor, então o valor de SNR é dado por

$$SNR(d, t) = \frac{P/A(d, f)}{N(f)\Delta f} \quad (5)$$

O valor de SNR é então utilizado para obter a taxa de erros de bits (BER, do inglês, Bit Error Rate). Utilizando a modulação BPSK (do inglês, *Binary Phase Shift Keying*), onde cada símbolo carrega um bit, o BER pode ser calculado através da fórmula [Rappaport et al. 1996]:

$$BER = \frac{1}{2} \left(1 - \sqrt{\frac{SNR(d, f)}{1 + SNR(d, f)}} \right) \quad (6)$$

Obtido o BER, a taxa de erros em pacotes (PER, do inglês Packet Error Rate) pode ser estimada como sendo complemento da probabilidade de que todos os bytes do pacote estejam corretos. Ou seja, sendo os pacotes formados por n bits, o PER é dado por

$$PER = 1 - (1 - BER)^n \quad (7)$$

4. Método Proposto

O primeiro passo do método de escalonamento proposto é obter a parte do espaço pelo qual cada nó é responsável. Diagramas Voronoi de cada uma das camada da rede são utilizados para isso. Neles, os sítios e as células representam, respectivamente, os nós da rede e suas áreas de monitoramento.

Após determinar a área pelo qual cada nó é responsável, o próximo passo consiste em verificar se existem nós que podem ser desativados temporariamente. A verificação é feita camada por camada, a começar por aquela mais próxima da superfície. Caso um nó possua uma área de monitoramento menor do que um dado limiar e sua ausência não desconecte alguma parte da rede, ele poderá ser desativado. Se isso ocorre, a região pelo qual ele era responsável é dividida entre seus vizinhos e o diagrama Voronoi da camada que o contém é atualizado. Quando não houver mais nós que possam ser desativados em uma camada, passa-se para a próxima. A verificação chega ao fim quando a camada mais profunda é atingida. Com isso, o escalonamento está feito.

Depois de realizado o escalonamento, a rede então se torna ativa e os nós começam a enviar dados para a superfície. Em algum momento, alguns nós da rede devem começar a ficar inacessíveis ao esgotarem suas fontes de energia. Assim que um nó da rede morre, sua área de monitoramento deve ser divida entre os outros ao seu redor. Para representar essa mudança, o diagrama de Voronoi da camada a qual ele pertence precisa ser atualizado. Também é preciso verificar se existem nós que perderam contato com aqueles mais próximos da superfície. Caso isso aconteça, pode ser necessário reativar algum dos nós previamente desativados. Essa necessidade pode ser verificada ao se refazer o diagrama de Voronoi da camada, considerando como sítios os nós que ainda estão ativos e aqueles que foram desativados. Vale notar que é possível que não existam alternativas para o desligamento de alguns nós. Por exemplo, um nó que é responsável sozinho por transmitir e repassar os dados de nós da sua camada e das outras abaixo, tem sua energia esgotada mais rapidamente. Ao morrer, faz com que todos os nós que dependem dele não consigam mais enviar dados para a superfície por não haver outra rota alternativa. Isso ajuda no processo de "morte"da rede. A rede é considerada morta quando apenas uma porcentagem pequena de seus nós ainda estão ativos e conseguem enviar mensagens para o nós da superfície.

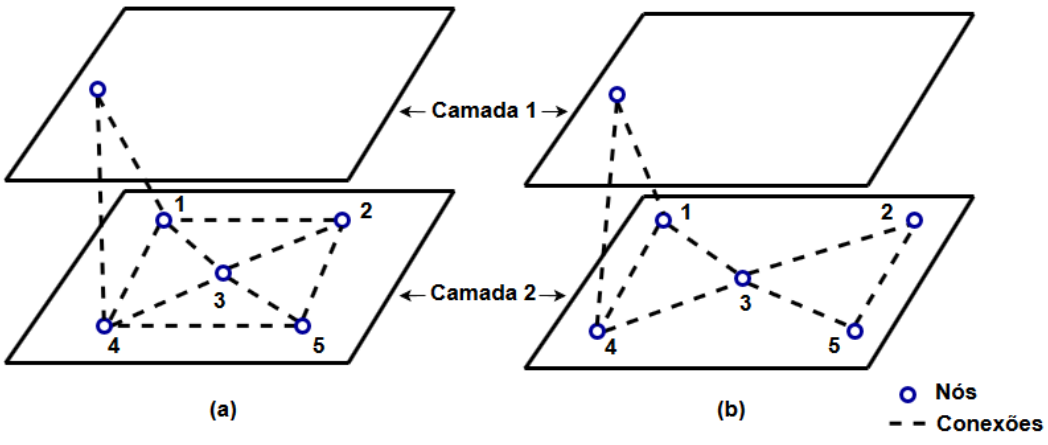
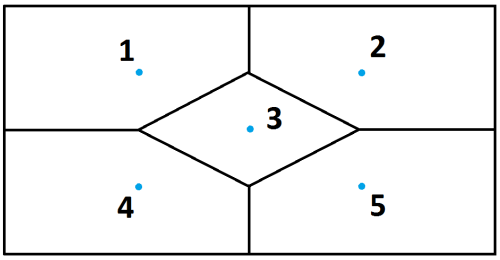


Figura 3. Exemplo de redes com topologias similares.

A parte de verificação do método é ilustrada utilizando as topologias mostradas nas Figuras 3(a) e 3(b). As duas topologias são similares, se diferenciando apenas em duas conexões: uma entre os nós 1 e 2 e outra entre os nós 4 e 5. Essas conexões existem na primeira, mas não na segunda. A verificação em ambos os casos começa pela camada 2, uma vez que a primeira possui somente 1 nó. Os diagramas de Voronoi dessa camada em ambas topologias são bastante semelhantes, sendo eles aproximadamente representados pela Figura 4(a). Supondo que a área de monitoramento do nó 3 seja menor do que um limiar previamente definido, deve-se verificar se ele pode ser desativado. Na primeira topologia isso é possível porque todos os nós que se comunicam com ele conseguem enviar dados para o nó da primeira camada através de outros. Já na segunda topologia, seu desligamento faria com que os nós 2 e 4 não conseguissem se comunicar com o restante da rede e por isso ele não pode ser desativado. No primeiro caso, após o nó ser desativado, o diagrama de Voronoi da segunda camada é atualizado e fica como aquele mostrado na Figura 4(b). Após esse passo, em ambos os casos não existe mais nenhum nó responsável por uma área muito pequena e logo termina-se a verificação.



(a) Diagrama de Voronoi inicial

(b) Diagrama de Voronoi atualizado

Figura 4. Exemplo de atualização de um diagrama de Voronoi quando um nó é desativado.

5. Avaliação de Desempenho

Nesta seção são descritas as configurações e os resultados dos experimentos realizados. A avaliação do método proposto foi feita através de simulações utilizando o software MATLAB.

5.1. Configurações dos Experimentos

A não ser quando descrito de outra forma, todos os experimentos foram realizados utilizando uma rede distribuída ao longo de cinco camadas quadradas de 1500x1500 m^2

cada. A primeira camada da rede sempre possui somente um nó, chamado de nó *sink*. Já cada uma das outras camadas possuem n nós cada. Varia-se n para se obter diversas configurações de densidade de nós. As profundidades das camadas são determinadas aleatoriamente, com a restrição de que nenhuma fique isolada. Dentro das camadas, os nós são distribuídos de forma aleatória. O limiar da área de monitoramento é definido como sendo uma porcentagem da área de transmissão dos nós. É considerado que a rede está morta quando 50% ou mais de seus nós não conseguem transmitir dados para o nó *sink*.

A troca de mensagens foi simulada através de um esquema onde somente um nó transmite a cada unidade de tempo. A escolha do nó que irá transmitir é feita da seguinte maneira:

1. Escolhe-se aleatoriamente um nó da rede, que não seja o da primeira camada, para enviar seus dados para a superfície.
2. Traça-se uma rota do nó escolhido até o nó *sink*.
3. O primeiro nó na rota realiza uma transmissão para o segundo. Caso haja sucesso, o segundo transmite uma confirmação de volta. Em seguida, o segundo repassa a mensagem para frente até que ela chegue no último nó da rota. Cada nó pode tentar transmitir a mesma mensagem por até 3 vezes.
4. Se o nó *sink* recebe os dados ou se as tentativas de transmissão são esgotadas, então começa-se o ciclo novamente na etapa 1.

A estimativa de erros em entregas de pacotes descrita na Seção 3.3 é utilizada para verificar o sucesso das transmissões. O coeficiente de espalhamento escolhido foi aquele que corresponde ao espalhamento prático (k = 1.5). A velocidade do vento e o nível de atividade de embarcações foram considerados como sendo nulos. É considerado que a taxa de transmissão da rede é de uma mensagem por segundo. Também assumiu-se que os transmissores possuíam potência de 1 W e transmitiam a 100 kHz. A menos que especificado com outro valor, o alcance dos transmissores é de 250 metros. Já a largura de banda de ruído dos receptores foi utilizada com um valor igual a 3 dB. O tamanho dos pacotes é de 500 bytes. Um resumo dos valores dos parâmetros utilizados se encontra na Tabela 1.

Parâmetro	Valor
Taxa de transmissão	1 nó/ segundo
Frequência de transmissão	100 kHz
Tamanho do pacote	500 bytes
Alcance de transmissão	250 metros
Potência de transmissão	1 W
Largura de banda de ruído	3 dB
Coeficiente de espalhamento	1.5 (espalhamento prático)
Velocidade do vento	0 m/s
Atividade de embarcações	0

Tabela 1. Parâmetros utilizados durante a simulação.

O consumo de energia da rede foi modelado da seguinte forma:

- A cada unidade do tempo, os nós consomem um pouco de sua energia. Nós ativos gastam uma quantidade maior de energia do que aqueles que estão desativados.

- A transmissão de dados consome energia somente do transmissor. Isso porque para receberem mensagens, os nós precisam escutar o ambiente o tempo todo.
- Os receptores gastam energia se estão ativos. Caso estejam desativados, não recebem mensagens.

Os valores associados aos consumos foram utilizados como percentuais nas simulações. Isso permite uma maior generalidade, uma vez que o consumo varia de dispositivo para dispositivo. Assim, cada nó da rede possui inicialmente 100% de sua capacidade energética. Também é considerado que a capacidade energética do nó *sink* é maior do que a dos demais. Isso se deve ao fato de que na superfície tem-se uma maior facilidade para obtenção de energia, podendo ser utilizada a energia solar para recarregá-lo, por exemplo. Os valores de consumo de energia utilizados se encontram na Tabela 2.

Estado	Consumo (em %)
Transmitindo	0.2
Ativado	0.01
Desativado	0.001

Tabela 2. Parâmetros de consumo de energia utilizados nos experimentos.

O tempo de vida da rede é medido baseado no número de mensagens trocadas. Isso porque, como dito anteriormente, foi considerado que é realizada uma transmissão por segundo.

Os resultados consistem da média dos valores obtidos através da execução de cada experimento 30 vezes. O intervalo de confiança utilizado é igual a 95%.

5.2. Resultados dos Experimentos

Primeiramente procurou-se verificar a influência do número de camadas na quantidade de nós desativados. Para isso, o número de nós da rede foi mantido em 200, o limiar da área de monitoramento foi fixado em 20% e variou-se o número de camadas até que não fosse

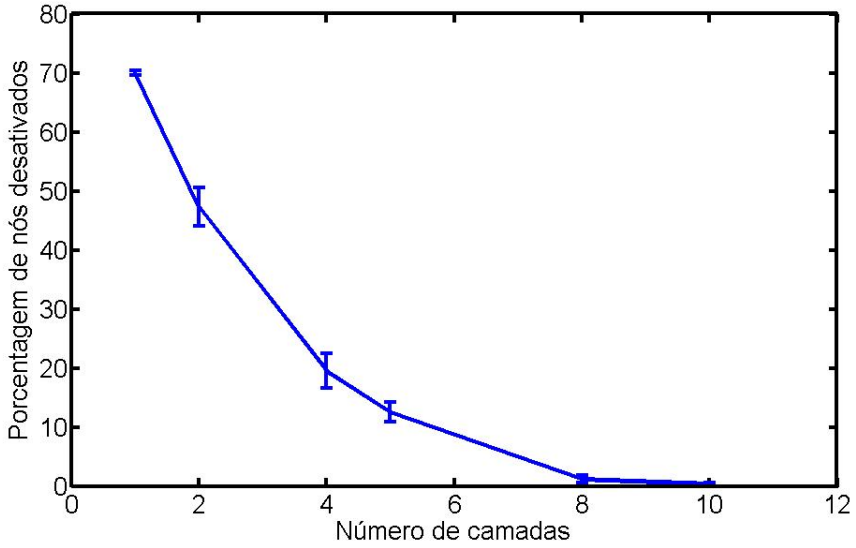
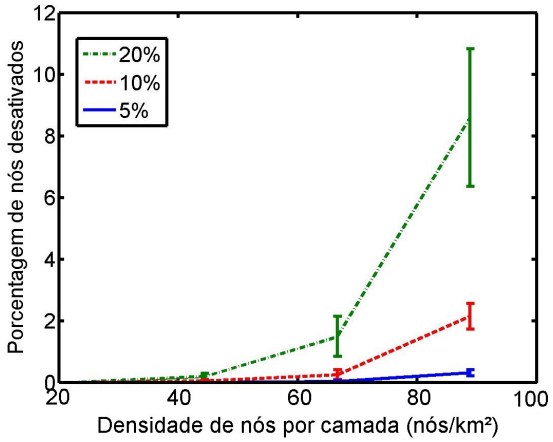
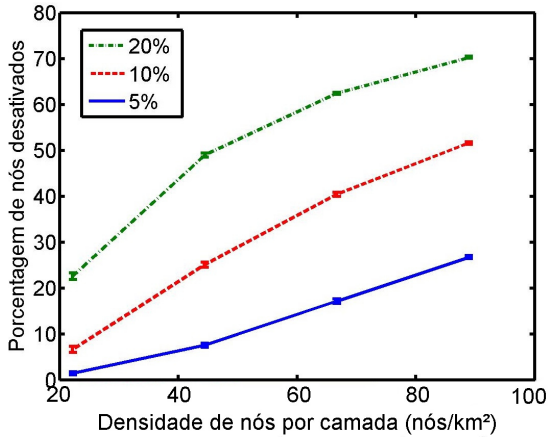


Figura 5. Porcentagem de nós desativados de acordo com o número de camadas da rede.

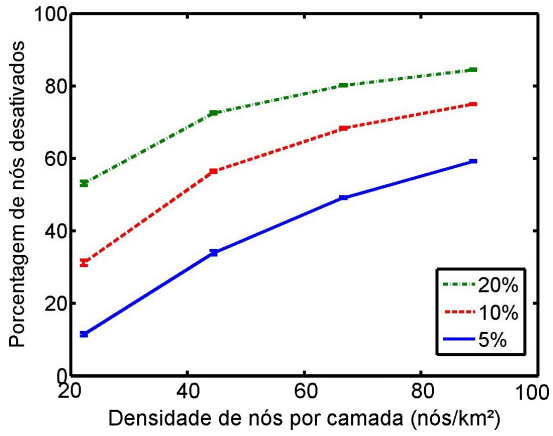
possível desativar algum nó. Como pode ser observado na Figura 5, o número de nós desativados cai rapidamente com o crescimento do número de camadas. Este comportamento era esperado, uma vez que a adição de uma nova camada aumenta a área de cobertura da rede e assim é necessário que mais nós fiquem ativos.



(a) Alcance de transmissão igual a 100 m.



(b) Alcance de transmissão igual a 250 m.



(c) Alcance de transmissão igual a 400 m.

Figura 6. Número de nós desativados por densidade de nós por camada, para três limiares de área de monitoramento, com diferentes valores de alcance de transmissão.

A Figura 6 mostra a relação entre a densidade de nós por camada e o número de nós desativados, para os limiares de área de monitoramento de 5%, 10% e 20%. O alcance de transmissão varia em cada uma das figuras, assumindo os valores de 100, 250 e 400 m nas Figuras 6(a), 6(b) e 6(c), respectivamente. É possível perceber que, como esperado, a porcentagem de nós desativados cresce a medida que a densidade de nós aumenta. Esse crescimento também pode ser observado quando se aumenta o valor do limiar da área de monitoramento. Os resultados também mostram que o alcance de transmissão impacta diretamente na quantidade de nós desativados, podendo fazer com que nenhum nó seja desativado, como mostrado na Figura 6(a), ou com que mais de 80% da rede o sejam, como na Figura 6(c).

Para verificar o tempo de vida da rede, voltou-se a considerar o limiar de área de monitoramento como 20%. A Figura 7 mostra a relação entre a densidade de nós por camada e o tempo de vida da rede, sem e com a utilização do método proposto. Nela é possível ver que a utilização do método proposto realmente leva a um aumento no tempo de vida da rede, principalmente quando se tem uma densidade de nós alta.

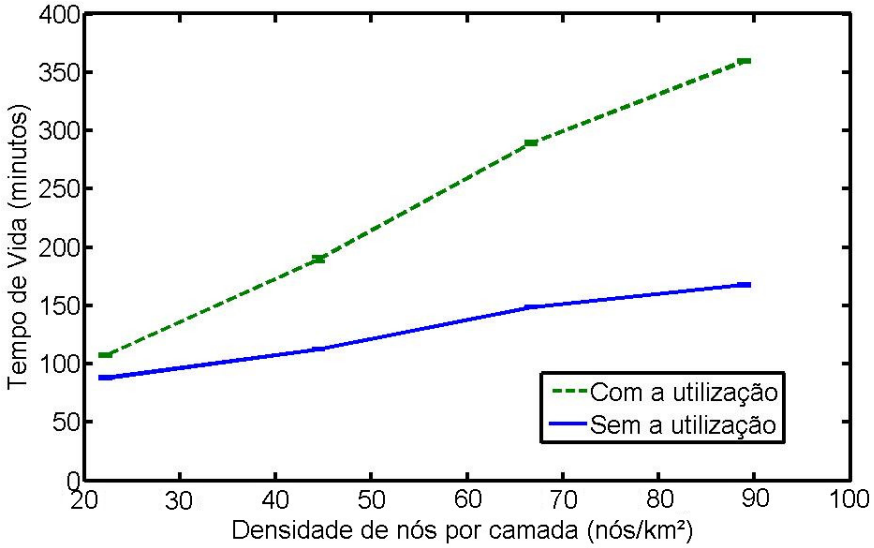


Figura 7. Tempo de vida de uma rede por densidade de nós por camada, com e sem o método proposto.

A Figura 8 mostra a relação entre o número de envio de dados até a nó *sink* e a densidade de nós por camada, com e sem a utilização do método proposto. Envios de dados ocorrem todas as vezes que se escolhe um nó para mandar seus dados para a superfície. Pode-se observar que, como consequência do aumento no tempo de vida da rede, a utilização do método proposto leva a um aumento no número de transmissões.

A relação entre o número médio de mensagens por envio de dados, variando-se a densidade de nós por camada, é mostrada na Figura 9. Percebe-se que utilizando o método proposto, o número médio de mensagens por envio é maior do que sem a sua utilização. Isso se deve principalmente ao fato de que desativar alguns nós faz com que a distância entre nós vizinhos seja maior. Com esse aumento, a probabilidade de erros em pacotes cresce e podem ser necessárias mais retransmissões para entregar os dados corretamente.

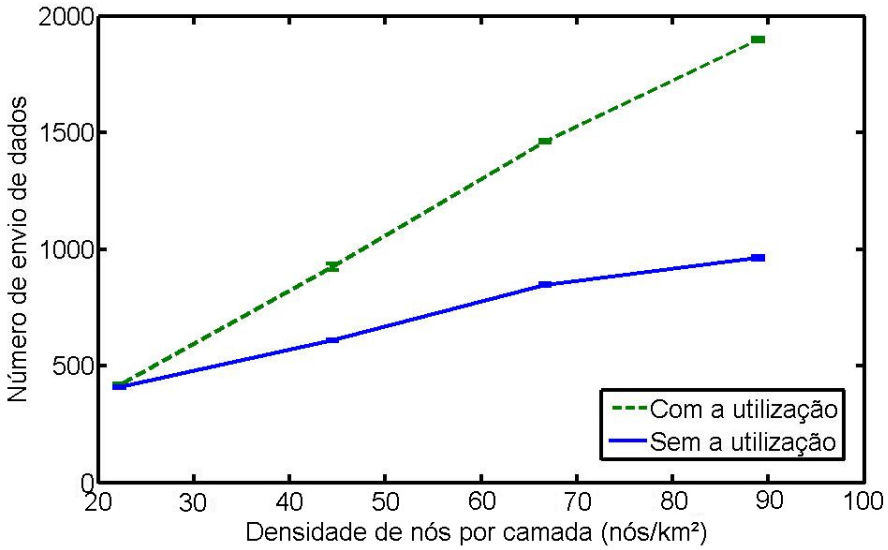


Figura 8. Número de envio de dados por densidade de nós por camada, com e sem o método proposto.

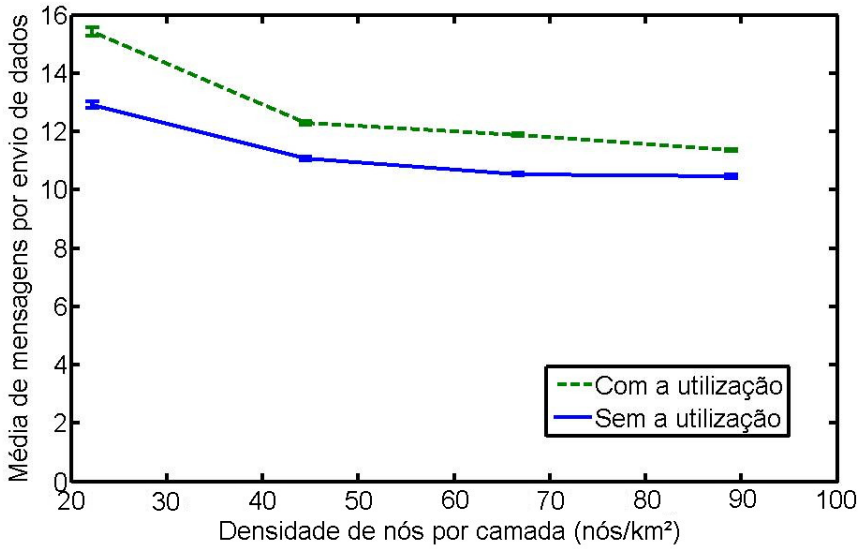


Figura 9. Número médio de mensagens por transmissão por densidade de nós por camada, com e sem o método proposto.

6. Conclusão e Trabalhos Futuros

Nesse artigo é apresentado um método para a realização do escalonamento de nós em redes de sensores aquáticas estratificadas. Este método utiliza diagramas de Voronoi em cada uma das camadas da rede para determinar quais nós podem ser desativados temporariamente. Isso possui o objetivo de poupar energia de tais nós para que eles possam ser utilizados em momentos posteriores. Um nó no entanto, só poderá ser desativado caso não desconecte outros nós da rede. Com isso esperava-se aumentar o tempo de vida da rede, ao mesmo tempo que mantém sua conectividade.

Simulações mostraram que o número de nós desativados cresce com a densidade de nós, com o alcance de transmissão e com o limiar da área de monitoramento. Elas também mostraram que, mesmo com um aumento no número médio de mensagens por envio de dados, o método proposto é capaz de cumprir seu objetivo de aumentar o tempo de vida da rede.

Como trabalhos futuros, pretende-se avaliar o uso do método quando se tem uma quantidade maior de tráfego de dados na rede. Também deverá ser avaliado o uso dele em redes aquáticas tridimensionais que não são estratificadas.

Referências

Akyildiz, I. F., Pompili, D., and Melodia, T. (2005). Underwater acoustic sensor networks: research challenges. *Ad Hoc Networks*, 3(3):257 – 279.

Brekhovskikh, L. M. (2003). *Fundamentals of ocean acoustics*. Springer Science & Business Media.

Coates, R. F. (1989). *Underwater acoustic systems*. Halsted Press.

Coutinho, R. W., Boukerche, A., Vieira, L. F., and Loureiro, A. A. (2015). Modeling and analysis of opportunistic routing in low duty-cycle underwater sensor networks. In *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 125–132. ACM.

Coutinho, R. W., Boukerche, A., Vieira, L. F., and Loureiro, A. A. (2016a). Geographic and opportunistic routing for underwater sensor networks. *IEEE Transactions on Computers*, 65(2):548–561.

Coutinho, R. W., Boukerche, A., Vieira, L. F., and Loureiro, A. A. (2016b). On the design of green protocols for underwater sensor networks. *IEEE Communications Magazine*, 54(10):67–73.

Coutinho, R. W., Boukerche, A. F., Vieira, L., and Loureiro, A. (2016c). A novel centrality metric for topology control in underwater sensor networks. In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 205–212. ACM.

Coutinho, R. W., Vieira, L. F. M., and Loureiro, A. A. F. (2013). DCR: Depth-controlled routing protocol for underwater sensor networks. In *2013 IEEE Symposium on Computers and Communications (ISCC)*, pages 453–458. IEEE.

Coutinho, R. W. L., Boukerche, A., Vieira, L. F. M., and Loureiro, A. A. F. (2014). GEDAR: geographic and opportunistic routing protocol with depth adjustment for mobile underwater sensor networks. In *2014 IEEE International Conference on Communications (ICC)*, pages 251–256. IEEE.

Coutinho, R. W. L., Boukerche, A., Vieira, L. F. M., and Loureiro, A. A. F. (2016d). Design guidelines for opportunistic routing in underwater networks. *IEEE Communications Magazine*, 54(2):40–48.

de Berg, M., Cheond, O., van Kreveld, M., and Overmars, M. (2008). *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd edition.

Erol, M., Vieira, L. F., Caruso, A., Paparella, F., Gerla, M., and Oktug, S. (2008). Multi stage underwater sensor localization using mobile beacons. In *Sensor Technologies and Applications, 2008. SENSORCOMM'08. Second International Conference on*, pages 710–714. IEEE.

Erol, M., Vieira, L. F., and Gerla, M. (2007a). Localization with Dive'N'Rise (DNR) beacons for underwater acoustic sensor networks. In *Proceedings of the second workshop on Underwater networks*, pages 97–100. ACM.

Erol, M., Vieira, L. F. M., and Gerla, M. (2007b). AUV-aided localization for underwater sensor networks. In *Wireless Algorithms, Systems and Applications, 2007. WASA 2007. International Conference on*, pages 44–54. IEEE.

Lee, U., Wang, P., Noh, Y., Vieira, L. F. M., Gerla, M., and Cui, J.-H. (2010). Pressure routing for underwater sensor networks. In *INFOCOM 2010. The 29th Conference on Computer Communications. IEEE*, pages 1676–1684.

Noh, Y., Lee, U., Lee, S., Wang, P., Vieira, L. F., Cui, J.-H., Gerla, M., and Kim, K. (2016). Hydrocast: pressure routing for underwater sensor networks. *IEEE Transactions on Vehicular Technology*, 65(1):333–347.

Rappaport, T. S. et al. (1996). *Wireless communications: principles and practice*, volume 2. Prentice Hall PTR New Jersey.

Stojanovic, M. (2007). On the relationship between capacity and distance in an underwater acoustic communication channel. *ACM SIGMOBILE Mobile Computing and Communications Review*, 11(4):34–43.

Vieira, L., Loureiro, A., Fernandes, A., and Campos, M. (2010). Redes de sensores aquáticas. *XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, Gramado, RS, Brasil*, 24.

Vieira, L. F. M. (2012). Performance and trade-offs of opportunistic routing in underwater networks. In *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2911–2915. IEEE.

Vieira, L. F. M., Kong, J., Lee, U., and Gerla, M. (2006). Analysis of aloha protocols for underwater acoustic sensor networks. *Extended abstract from WUWNet*.

Vieira, M., Vieira, L., Ruiz, L. B., Loureiro, A. A. F., Fernandes, A. O., and Nogueira, J. M. S. (2003). Scheduling nodes in wireless sensor networks: A voronoi approach. In *Local Computer Networks, 2003. LCN'03. Proceedings. 28th Annual IEEE International Conference on*, pages 423–429. IEEE.

Wang, Z. and Wang, B. (2016). A novel node sinking algorithm for 3d coverage and connectivity in underwater sensor networks. *Ad Hoc Networks*.

Wu, J., Wang, Y., and Liu, L. (2013). A voronoi-based depth-adjustment scheme for underwater wireless sensor networks. *Int. J. Smart Sens. Intell. Syst*, 6:244–258.

Um Sistema de Identificação Antecipada e Transmissão Prioritária de Alertas Médicos sobre WBAN e WLAN

Andressa Vergütz[1], Rafael da Silva[1], Alex B. Vieira[2], Michele Nogueira[1]

[1]Depto. de Informática – NR2 – Universidade Federal do Paraná (UFPR)
[2]Depto. de Ciência da Computação – Universidade Federal de Juiz de Fora (UFJF)

{avergutz,rasilva,michele}@inf.ufpr.br, alex.borges@ufjf.edu.br

Abstract. *The anticipated identification of critical events in patients' health reaches its paramount importance when associated to the urgent delivery of the alerts to healthcare professionals. However, wireless networks suffer from communication constraints, being vulnerable to interferences and losses. Current mechanisms in WBANs and WLANs either are concerned with the detection of critical events or with the priority on data transmission, inexisting solutions to jointly address both aspects. This work presents SANTE, the first* **S***ystem for* **A***nticipated identificatio***N** *and* **T***ransmission of m***E***dical alerts on WBAN/WLAN. Particularly, the system foresees trends about the imminence of critical events on patients' vital signs through a set of statistical indicators. When those trends are identified, SANTE generates medical alerts. It assigns the highest level of priority to those alerts and offers priority on medium access by reducing contention window and AIFS range on WLAN. Even in a dense scenario, simulation results show in average a reduction of 39% in the medical alerts transmission latency and 8% in their losses.*

Resumo. *A identificação antecipada de eventos críticos na saúde dos pacientes atinge seu ápice quando associada à entrega imediata dos alertas aos profissionais da saúde. Entretanto, as redes sem fio apresentam limitações na comunicação, sendo vulneráveis a interferências e perdas. As soluções existentes em redes corporais sem fio (WBANs) e redes locais sem fio (WLANs) ora se preocupam com a detecção de eventos críticos, ora com a transmissão prioritária de dados, sendo inexistentes soluções abordando os dois aspectos juntos. Este trabalho apresenta SANTE (do inglês,* **S***ystem for* **A***nticipated identificatio***N** *and* **T***ransmission of m***E***dical alerts on WBAN/WLAN), o primeiro sistema de identificação antecipada e transmissão prioritária de alertas médicos no contexto híbrido WBAN/WLAN. SANTE prediz tendências sobre a iminência de eventos críticos no comportamento dos sinais vitais por meio de um conjunto de indicadores estatísticos. Identificadas essas tendências, SANTE gera alertas médicos que recebem o maior nível de prioridade para transmissão e ele oferece prioridade de acesso ao meio através da redução da janela de contenção e intervalo AIFS na WLAN. Os resultados de simulação mostram que, mesmo em um cenário denso, SANTE apresenta uma redução média de 39% na latência de transmissão dos alertas e 8% na taxa de perda.*

1. Introdução

O rápido avanço nas tecnologias de comunicação sem fio e nos nanossistemas tem contribuído com o desenvolvimento de sensores inteligentes implantáveis ou vestíveis no

corpo humano. A união desses sensores e das redes de comunicação, conhecida por Redes Corporais sem Fio (WBANs), permite a coleta e o monitoramento de sinais vitais [Movassaghi et al. 2014]. As WBANs são bem aplicadas em um contexto médico, onde aplicações recebem os dados coletados por sensores e os informam aos profissionais ou centros de saúde. Um benefício desse acompanhamento contínuo é a possível identificação de riscos e problemas de saúde em estágios iniciais [Latré et al. 2011], promovendo cada vez mais o uso dessas aplicações médicas [Cavallari et al. 2014].

Entretanto, avanços ainda precisam ser feitos para a completa integração das aplicações médicas com as redes sem fio, haja visto que essas aplicações carregam dados de extrema importância. Por exemplo, a análise de dados coletados por sensores auxilia na geração de alertas médicos sobre eventos que venham a ameaçar a vida dos pacientes, tais como ataques cardíacos (i.e., eventos críticos) [Cavallari et al. 2014]. Porém, a identificação dos eventos críticos para emissão dos alertas ocorre quando estes eventos já estão em progresso. Também, por sua natureza emergencial, esses alertas requerem a transmissão imediata, aceitando no máximo 125 ms de latência [Association et al. 2012]. Esse nível de latência impulsiona melhorias nas redes sem fio, uma vez que desde a aquisição dos dados pelos sensores até a sua entrega no destino final, diversas redes e tecnologias são utilizadas. Além disso, apesar de se esperar que as WBANs sejam dedicadas à aplicação médica, a infraestrutura de um ambiente médico pode carregar diversos fluxos de dados, como voz e vídeo. A competição pelo meio de transmissão causa atrasos aos alertas médicos [Movassaghi et al. 2014], com possíveis consequências desastrosas à saúde do paciente. Desse modo, além de identificar o quanto antes os eventos críticos, se faz necessária a transmissão imediata desses alertas. Assim, os profissionais da saúde poderão tomar ações a tempo de salvar vidas.

Alguns estudos propõem formas de identificar eventos críticos e gerar alertas médicos, além de classificá-los em níveis de prioridade [Misra and Sarkar 2015, Kim and Kim 2015, Kathuria and Gambhir 2016]. Outros estudos propõem mecanismos para a transmissão imediata dos alertas [Gündoğdu and Çalhan 2016, Bhandari and Moh 2016]. No geral, a identificação dos eventos críticos delimita-se a comparar limiares predefinidos para sinais vitais específicos. Além disso, esses trabalhos ignoram os outros fluxos de dados presentes nas redes sem fio. Ainda, poucos trabalhos consideram a integração WBAN e WLAN e seus fluxos de dados [Bradai et al. 2016, Rashwand and Misic 2015]. Entretanto, quando considerado essa integração, esses trabalhos classificam os alertas com a mesma prioridade do tráfego de voz de uma WLAN, não oferecendo o tratamento necessário. Em suma, esses estudos ora se preocupam em identificar eventos críticos, ora em transmitir prioritariamente os alertas médicos gerados. Sendo, no melhor do nosso conhecimento, inexistentes estudos que realizam tanto a identificação quanto a transmissão prioritária dos alertas no contexto WBAN/WLAN.

Nesse sentido, este trabalho apresenta o SANTE (do inglês, **S***ystem for* **A***nticipated Identificatio***N** *and* **T***ransmission of M***E***dical Alerts on WBAN/WLAN*), o primeiro sistema para identificação antecipada e transmissão prioritária de alertas médicos nas WBANs integradas às WLANs. Particularmente, o sistema tem como objetivo predizer tendências no comportamento dos sinais vitais sobre a iminência de eventos críticos por meio de um conjunto genérico de indicadores estatísticos. Esses indicadores predizem a ocorrência de eventos críticos, como ataques cardíacos e paradas respiratórias, funda-

mentados na análise de comportamentos genéricos [Dakos et al. 2012]. Quando previsto algum evento crítico, o sistema gera alertas médicos e atribui aos mesmos o maior nível de prioridade de acesso ao meio da WLAN, através de uma nova categoria de acesso exclusiva para os alertas. Desse modo, o sistema oferece um tratamento adequado aos dados emergenciais e permite que os profissionais da saúde reajam em tempo hábil.

Nós avaliamos a eficiência da proposta desse trabalho em duas fases. Inicialmente, na ferramenta R, nós realizamos a identificação dos eventos críticos, onde aplicamos os indicadores estatísticos sobre traços reais de frequência respiratória de um paciente de 70 anos com edema pulmonar [Dakos et al. 2012]. A seguir, avaliamos o desempenho do SANTE no Simulador de Redes NS-3, considerando cenários de ambientes domésticos densos e não densos. A partir do resultado da identificação de evento crítico, criamos o mapeamento de prioridades no NS-3 levando em conta diferentes fluxos de dados e, emitimos alertas médicos nos momentos em que os resultados estatísticos apontam eventos críticos. Na transmissão prioritária, empregamos uma nova categoria de acesso, exclusiva para os alertas médicos, e priorizamos o acesso ao meio na WLAN através da redução da janela de contenção e do intervalo AIFS. Os resultados apontam a viabilidade dos indicadores para predição de eventos críticos em dados vitais empíricos, identificando o evento crítico alvo com 12 minutos de antecedência. SANTE é capaz de reduzir em 39% o atraso nas transmissões de alertas médicos além de reduzir significativamente as perdas.

O restante deste artigo está organizado como segue. A Seção 2 apresenta os trabalhos relacionados à identificação e à transmissão prioritária de alertas médicos. A Seção 3 detalha o sistema SANTE. A Seção 4 descreve a avaliação de desempenho juntamente com os resultados obtidos. Por fim, a Seção 5 apresenta as conclusões e direções futuras.

2. Trabalhos Relacionados

Diferentes trabalhos propuseram tanto a identificação de eventos críticos quanto a classificação de alertas médicos em níveis de prioridades [Misra and Sarkar 2015, Kim and Kim 2015, Kathuria and Gambhir 2016]. Em [Misra and Sarkar 2015], os autores classificam os pacotes de dados conforme o seu índice de prioridade criado segundo características do paciente, como idade, sexo e doença. No estudo de [Kim and Kim 2015], os níveis de prioridades são associados ao tipo do sensor, considerando apenas os sensores ECG (frequência cardíaca), EEG (atividade cerebral) e EMG (atividade muscular), do nível mais alto para o mais baixo de prioridade. A solução proposta em [Kathuria and Gambhir 2016] utiliza algoritmos de aprendizagem de máquina, como *Support Vector Machine*, para classificar os pacotes com base nos atributos dos seus cabeçalhos, como tipo de tráfego e índice de alerta. Esse índice é definido segundo limiares do sinal vital recebido. No entanto, o uso de limiares específicos de sinais vitais para a identificação de um evento crítico requer conhecimento específico sobre cada sinal vital monitorado e ainda, identifica os eventos críticos quando estes já estão em progresso. Entretanto, todos esses estudos apenas consideram limiares dos sinais na identificação e ignoram os outros fluxos de dados, como voz e vídeo, na priorização do acesso ao meio.

A fim de transmitir imediatamente os dados emergenciais sobre a saúde dos pacientes, os estudos de [Gündoğdu and Çalhan 2016, Bhandari and Moh 2016] propõem mecanismos de transmissão prioritária. O trabalho de [Gündoğdu and Çalhan 2016] propõe o uso de filas não preemptivas no coordenador WBAN (ex. dispositivo móvel

do paciente) para transmissão empregando três níveis de prioridade: dados emergenciais (ECG), sob demanda (requisições específicas de sinais) e normal (sinais periódicos). Em [Bhandari and Moh 2016], os autores classificam os diferentes fluxos de tráfegos nas categorias emergencial (alertas médicos), sob demanda (sinais contínuos, como EEG), normal (sinais não contínuos, como temperatura) e tráfego não médico (áudio, vídeo, dados). Na transmissão prioritária, os autores dividem a fase de alocação de canal da WBAN em subfases exclusivas para cada prioridade, permitindo apenas que o tráfego emergencial utilize todas as subfases durante sua transmissão. Apesar de ambos os trabalhos alcançarem latências menores, a identificação do evento crítico ocorre pela comparação de limiares particulares dos sinais vitais para tomada de decisões, necessitando de conhecimentos específicos dos sinais. Esses trabalhos também não consideram a integração das diferentes redes sem fio e seus níveis de prioridades na transmissão.

Com o intuito de solucionar o desafio de mapeamento prioritário da WBAN dentro das categorias de acesso da WLAN, os trabalhos [Bradai et al. 2015, Rashwand and Misic 2015] propõem mecanismos para mapear as oito prioridades da WBAN definidas pelo TG6 da IEEE (*Task Group IEEE 802.15.6*) em quatro categorias de acesso das redes 802.11. [Rashwand and Misic 2015] mapeiam os dois maiores níveis de prioridade da WBAN, que engloba alertas médicos e dados de controle da rede, em um nível mais alto da categoria de acesso da WLAN (compreendendo também os dados de voz). Em [Bradai et al. 2016], os autores definem três categorias prioritárias de tráfego WBAN: emergencial, sob demanda e normal. Na sequência, os autores agregam os oito níveis de prioridades nessas categorias conforme similaridades, sendo que a categoria emergencial recebe prioridade de dados de voz. Ambos os trabalhos apresentam latências menores que um segundo, porém não tratam a detecção de eventos críticos. Além disso, eles priorizam os alertas médicos tal qual tráfego de voz na WLAN e por isso não garantem o tratamento especial que os alertas médicos necessitam. Assim, no melhor do nosso conhecimento, não existem trabalhos que realizam a identificação antecipada de eventos críticos e a transmissão prioritária de alertas médicos no contexto WBAN/WLAN.

3. O Sistema SANTE

Esta seção detalha o sistema SANTE, o primeiro sistema para identificação antecipada e transmissão prioritária de alertas médicos no contexto WBAN/WLAN. Os objetivos do sistema englobam a identificação antecipada de eventos considerados críticos em dados fisiológicos com base em mudanças significativas no comportamento dos mesmos e a transmissão prioritária dos alertas médicos. Tanto a identificação quanto a transmissão têm como objetivo informar os profissionais de saúde o mais rápido possível sobre eventos críticos identificados e, assim, auxiliar na ação rápida desses profissionais.

No escopo deste trabalho, os eventos críticos compreendem situações que podem resultar consequências grave na vida dos pacientes, como um ataque cardíaco ou uma parada respiratória. A gravidade de tais eventos demanda ações imediatas dos médicos e assim, atrasos podem ter consequências sérias. Dessa forma, a possibilidade de predizer a iminência de tais eventos e transmitir imediatamente alertas sobre eles, beneficia a monitoração médica, prevenindo exacerbações de doenças.

Para garantir a identificação antecipada de eventos críticos e a transmissão imediata dos alertas médicos sobre tais eventos nas WBANs e WLANs, a arquitetura geral do

sistema SANTE divide-se em quatro etapas: a *coleta dos dados*, a *predição de eventos críticos*, a *transmissão prioritária dos alertas médicos* e o *envio ao destino final*, ilustradas na Figura 1. A primeira etapa se resume na coleta dos sinais vitais pelos sensores corporais e o envio dos mesmos para o coordenador da WBAN (i.e. dispositivo móvel do paciente). A segunda etapa consiste na predição de tendências dos eventos críticos a partir de um conjunto de indicadores estatísticos genéricos a fim de avaliar o comportamento dos sinais vitais de pacientes. Quando os indicadores apontam a iminência de eventos críticos, o sistema gera um alerta médico. Na terceira etapa, realiza-se a classificação prioritária dos alertas médicos sobre todos os demais tipos de tráfego da rede. Com base na classificação de tráfego existente no padrão IEEE 802.11, propomos uma nova categoria de acesso ao meio exclusiva para os alertas médicos, a fim de transmitir prioritariamente os mesmos via WiFi. Por fim, a quarta etapa encaminha os dados para o destino final, nesse caso para o profissional da saúde e/ou servidor do hospital. As próximas subseções descrevem detalhadamente estas quatro etapas.

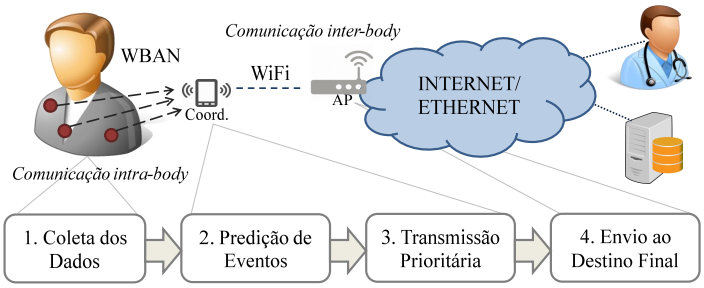


Figura 1. Arquitetura Geral do SANTE

3.1. Coleta dos Dados

Nesta etapa, os dados sobre os sinais vitais do paciente são coletados pelos sensores corporais e são enviados ao coordenador da WBAN. Os sensores se comunicam com o coordenador da WBAN por meio de tecnologias de comunicação de baixo alcance, comumente denominada de comunicação *intra-body*. Conforme ilustra a Figura 1, os dados coletados pelos sensores são transmitidos ao coordenador da WBAN. O coordenador então processa os dados e posteriormente os transmite aos centros de saúde, via Internet ou pela própria Ethernet do hospital [Movassaghi et al. 2014].

Existem diferentes tipos de sensores corporais, onde cada um monitora um respectivo sinal vital [Movassaghi et al. 2014]. Por exemplo, os sensores de eletrocardiograma (ECG) coletam dados sobre os batimentos cardíacos, enquanto o sensor de temperatura monitora a temperatura do corpo. Dessa maneira, vários sensores atuam ao mesmo tempo no corpo humano e diferentes tipos de dados corporais são coletados. Dentre esses dados, alguns possuem uma maior urgência para entrega aos profissionais de saúde (ex. ECG), enquanto outros englobam dados periódicos de rotina médica que não possuem tamanha criticidade [Cavallari et al. 2014]. Existem também os dados não médicos presentes no coordenador WBAN, como voz e vídeo. Além dos dados corriqueiros, há os alertas médicos. Estes requerem tratamento prioritário devido à sua natureza emergencial. De fato, os alertas podem informar os médicos sobre eventos críticos que venham ameaçar a vida dos pacientes. Quando antes as tendências desses eventos sejam identificadas e transmitidas ao centro médico, as chances de sobrevivência dos pacientes aumentam.

3.2. Predição de Eventos Críticos

Esta etapa aponta tendências sobre a iminência de situações emergenciais na saúde de pacientes e gera alertas médicos. A identificação antecipada de situações críticas na saúde toma como base um conjunto de indicadores estatísticos genéricos fundamentados em resultados de pesquisas científica realizadas nos últimos anos sobre predição de mudanças abruptas (também chamadas de transições críticas) [Dakos et al. 2012]. As doenças crônicas e depressão apresentam as transições críticas com episódios de ataques sem sintomas prévios aparentes, tais como arritmia cardíaca, ataques epilépticos, desordem bipolar e enxaqueca [Rikkert et al. 2016]. As transições críticas se caracterizam por uma alteração brusca e repentina de um estado para o outro. Elas forçam o estado da doença (conjunto de valores, como a frequência dos batimentos cardíacos) ultrapassar de forma repentina fronteiras críticas, sendo interpretados como mudanças abruptas no estado da saúde do paciente [Scheffer et al. 2009].

Um conjunto de indicadores estatísticos foram definidos para indicar transições críticas por [Scheffer et al. 2009] em diversos sistemas, tais como clima, mercado financeiro e doenças humanas. Neste trabalho, o sistema SANTE toma como base esses indicadores para avaliar as situações na saúde dos pacientes tendo como entrada dados coletados sobre a frequência respiratória. Esses indicadores apontam um evento crítico na saúde dos pacientes quando um conjunto de tendências específicas é encontrado nas curvas resultantes de um curto histórico desses indicadores. A relação entre essas tendências (detalhadas ainda nesta subseção) com as transições críticas foi estabelecida na literatura [Dakos et al. 2012]. Quando tendências de um evento crítico são identificadas, o sistema proposto gera um alerta médico, i.e. um pacote com prioridade de acesso ao meio. Este alerta que precisa ser enviado a centros médicos a fim de precaver os profissionais sobre a possível necessidade de atendimento emergencial ao paciente.

Particularmente, os indicadores utilizados conjuntamente neste trabalho consistem em: taxa de retorno, autocorrelação, variância, assimetria e curtose. A **taxa de retorno** representa o tempo levado durante a recuperação do estado do paciente após perturbações, tais como movimentos do corpo e frequência cardíaca elevada. O tempo de recuperação aumenta conforme a situação crítica se aproxima. Assim, uma taxa de retorno lenta indica uma possível direção a um evento crítico. Um aumento na **autocorrelação**, com curta defasagem no tempo, é esperado nas observações na iminência de um evento crítico. A autocorrelação tende a aumentar conforme o estado da doença se aproxima das fronteiras críticas, indicando que o estado do paciente se tornou mais similar entre observações consecutivas [Dakos et al. 2012]. Por exemplo, uma pessoa em estágios iniciais de depressão, com o estado da depressão próximo à transição crítica, tem uma recuperação de humor lenta após perturbações estressantes. Isto resulta em um aumento na autocorrelação do estado de humor nos momentos subsequentes [van de Leemput et al. 2014].

O retorno lento à estabilidade próximo a uma transição crítica faz com que os estados do paciente (ex.: os valores de batimentos cardíacos) alterem amplamente em torno do estado estável, causando um aumento na **variância** das observações. A variância consiste do segundo momento em torno da média de uma distribuição, estimando-a a partir da função de desvio padrão $SD = \frac{1}{n-1}\sum_{n}^{t=1}(z_t - \mu)^2$, onde μ é a média e z a variável analisada. Além disso, a presença de valores distantes no estado crítico provoca uma **assimetria** na curva da distribuição das observações. A assimetria pode aumentar ou

diminuir dependendo se a transição crítica do estado do paciente tende para um estado maior ou menor em relação ao estado atual. Quando o estado do paciente alcança valores extremos, próximo a uma transição crítica devido a fortes perturbações, a medida de **curtose** aumenta. A curtose torna-se leptocúrtica, isto é, ela apresenta um aumento na cauda da distribuição das observações [Dakos et al. 2012]. Esse comportamento indica que há a presença de valores raros nas observações dos estados e que o estado do paciente alcançou valores extremos, sinalizando indícios de evento crítico próximo.

Esses indicadores apresentam um comportamento genérico quando o estado do paciente se encontra próximo a uma transição crítica, sendo um aumento na autocorrelação, variância, assimetria e curtose, enquanto mostra uma queda na taxa de retorno [Dakos et al. 2012]. Esse conjunto de comportamentos para os indicadores acima aponta a proximidade das fronteiras críticas, e por consequência, a possibilidade do estado da paciente alterar para um estado crítico. Devido a isso, a identificação dos eventos críticos sobre sinais fisiológicos se embasou nesse conjunto de comportamentos. Em suma, quando esse conjunto de comportamentos é identificado, gera-se um alerta médico.

3.3. Transmissão Prioritária dos Alertas Médicos

Uma vez identificado o evento crítico no coordenador da WBAN, um alerta médico é emitido para o centro de saúde. Por ser crítico, este alerta deve ser transmitido de forma prioritária, sofrendo o mínimo de interferência e atraso. Contudo, prover este requisito de qualidade (mínima interferência e atraso) na comunicação entre o coordenador WBAN e o seu ponto de acesso pode ser desafiador. As redes locais sem fio baseadas no padrão 802.11 têm apresentado desempenho insatisfatório à medida que a quantidade de usuários aumenta. Há uma ampla prevalência deste padrão nas WLANs, incluindo hospitais e clínicas em que o coordenador WBAN obtém o seu acesso através de uma ponte entre estas redes. Nos últimos anos, o crescimento observado na quantidade de dispositivos tem resultado na sobrecarga das WLANs, principalmente aquelas operando nas faixas livres em frequência de 2,4GHz e instaladas sem planejamento. Assim, falhas decorrentes desta sobrecarga nas WLANs podem resultar na perda dos alertas médicos.

Neste trabalho criamos uma nova categoria de acesso ao meio exclusiva para os alertas médicos com base em [Silva et al. 2016]. Esta categoria conta com um intervalo arbitrário entre quadros (AIFS) menor do que o empregado para categoria de voz. Além disso, a janela de contenção é reduzida a um único *slot*. Com isto, as transmissões de alertas do coordenador WBAN passam a ter prioridade sobre os demais tipos de tráfego. A Figura 2 ilustra a categoria de acesso para o alerta médico em relação às demais categorias.

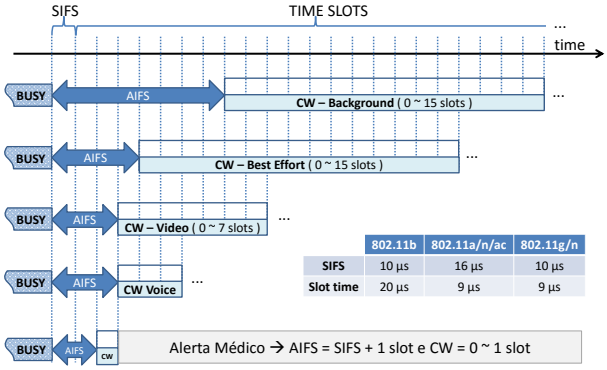


Figura 2. As transmissões iniciam dentro da janela de contenção (CW)

O uso de um intervalo AIFS tão pequeno, praticamente sem contenção, pode ser um problema se houver um grande número de alertas. Todavia, alguns ajustes são feitos para evitar que isto ocorra ou que gere um impasse ou bloqueio do mecanismo. Primeiro, limitamos o tamanho dos quadros de alertas críticos em 50 bytes, minimizando o impacto de colisões, pois quadros menores ocupam menos o canal e assim, caso ocorra uma colisão, o desperdício é minimizado. Segundo, controlamos o fluxo dos alertas dos dispositivos móveis, evitando rajadas que enfileiram os quadros na camada MAC, promovendo uma melhor distribuição desses fluxos. Terceiro, paramos a transmissão dos alertas tão logo sejam recebidos pelo sistema do centro médico, evitando retransmissões desnecessárias. Por último, emitimos alertas nesta categoria apenas para eventos críticos. Com estes cuidados, garante-se uma via quase exclusiva para os alertas médicos em seu caminho entre o coordenador WBAN e o ponto de acesso da WLAN. Uma vez no ponto de acesso, a transmissão do alerta pode contar com redes cabeadas onde podem ser aplicadas reservas de banda ou melhores técnicas de priorização.

3.4. Envio ao Destino Final

A transmissão dos alertas médicos do ponto de acesso até o servidor do hospital emprega uma rede cabeada ponto a ponto do próprio hospital. No geral, redes cabeadas costumam contar com diferentes tipos de serviços, dependendo do ambiente de atuação. Assim, consideramos que a rede do hospital ou a rede doméstica possui técnicas de priorização, como DiffService e LLQ, onde pacotes com o mesmo rótulo e tipo de serviço recebem o mesmo tipo de tratamento [Cisco 2006]. Desta forma, os pacotes de alertas médicos devem receber o maior nível de prioridade para transmissão a fim de garantir a latência máxima suportada pelas aplicações médicas. Para tanto, espera-se que o centro de saúde ofereça suporte adequado para o tratamento de alertas médicos das WBANs, uma vez que eles envolvem a vida do paciente, de forma que não comprometa o diagnóstico e o monitoramento médico, e ainda, o funcionamento das outras etapas do SANTE.

4. Avaliação de Desempenho

Avaliamos o sistema SANTE através do R[1] e simulações considerando um cenário WBAN/WiFi no NS-3[2]. Particularmente, tomamos como base no R um conjunto de indicadores sobre sinais vitais reais e assim, identificamos a iminência de eventos críticos. Quando identificado um evento crítico, geramos alertas médicos que servem de entrada para as simulações no NS-3. No simulador, classificamos esses alertas como prioritários e os transmitimos seguindo um acesso ao meio prioritário para esses alertas. Nós, então, avaliamos o atraso médio e a taxa de perda seguindo as especificações do SANTE. Comparamos os resultados obtidos aos observados quando utilizamos sistemas tradicionais, sem identificação e priorização de alertas médicos. As próximas subseções descrevem a metodologia de avaliação e apresentam os resultados observados das simulações.

4.1. Metodologia de Avaliação

Nós criamos um cenário de simulação no NS-3 próximo aos ambientes domésticos. Este cenário conta com apenas um ponto de acesso (AP) e várias pessoas utilizando seus dispositivos móveis. Dentre essas pessoas, algumas são monitoradas por aplicações médicas

[1]R, https://www.rstudio.com/. Último acesso em Dez/2016.

[2]Network Simulator NS-3, https://www.nsnam.org/. Último acesso em Dez/2016.

que geram alertas. Neste cenário, o AP centralizado possui duas interfaces de rede. Ele recebe dados dos dispositivos móveis via WiFi (802.11n) e transmite os alertas médicos para o destino final, considerado neste caso o servidor do hospital, via Ethernet. Em um ambiente mais realístico, haveria o AP da rede doméstica e o AP do hospital, porém reduzimos o escopo da simulação para apenas um AP.

Os dispositivos móveis consistem de *smartphones* que possuem aplicações médicas ou aplicações gerais, ambas gerando fluxos de dados UDP. Os dispositivos com aplicações médicas representam os coordenadores da WBAN e enviam apenas alertas médicos a cada 1,5 segundos para o servidor do hospital. Os dispositivos móveis com aplicações gerais enviam dados normais de rede, voz e vídeo para o AP, com fluxos de dados do tipo ON/OFF. Os fluxos de dados de voz e vídeo são constantes conforme aplicações de videoconferências. Enquanto, os fluxos de dados normais são gerados em rajadas de acordo com uma distribuição de Pareto, similar aos que representam o tráfego Web predominante em *hotspots* públicos [Divgi and Chlebus 2013].

A quantidade de dispositivos móveis com aplicações gerais variou a cada amostragem com duração de 13 segundos, iniciando com 5 dispositivos móveis e encerrando com 35. Esse processo foi repetido 35 vezes, reiniciando os contadores e o gerador de números aleatórios do simulador a cada amostragem. Em todas as execuções, além dos dispositivos móveis com aplicações gerais, há 10 dispositivos com aplicações médicas emitindo alertas aleatórios. As simulações foram realizadas com dois diferentes cenários: o primeiro empregando um sistema tradicional sem identificação e priorização de alertas médicos, e o segundo utilizando o sistema proposto.

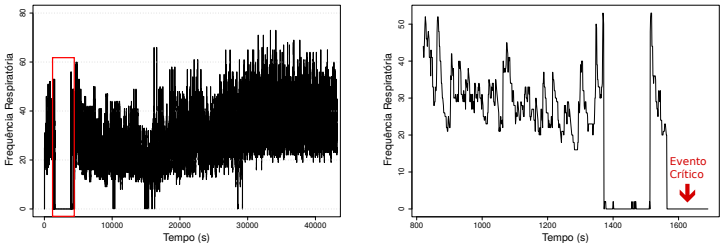
Todas as análises foram realizadas sobre um contexto de canal saturado com 100 Mbps de fluxos de dados normais de rede para sobrecarregar o canal WiFi, a fim de testar o envio prioritário dos alertas médicos em situações de sobrecarga no ambiente. A Tabela 1 apresenta os parâmetros utilizados em cada fluxo de tráfego. Os fluxos de vídeo apresentam taxas de dados de 384 Kbps, tal qual um fluxo de vídeo durante videoconferências. Os pacotes de alertas médicos receberam valores de tamanho e taxa de dados similar as WBANs [Bhandari and Moh 2016], e os pacotes de voz seguiram valores semelhantes as aplicações de áudio [IEEE 2012]. Os diferentes tipos de tráfegos foram marcados com níveis de prioridades conforme as categorias de acesso do WiFi, seguindo a ordem de prioridade de voz, vídeo e dados normais. Os alertas médicos recebem o nível mais alto de prioridade, sendo rotulados como sendo da categoria de acesso AC_AM.

Tabela 1. Parâmetros utilizados nas simulações

Tráfego	Prioridade	Tamanho do Pacote	Taxa de Dados
Normal de rede	AC_BE	1450 bytes	100 Mbps
Vídeo	AC_VI	1316 bytes	384 Kbps
Voz	AC_VO	64 bytes	64 Kbps
Alerta Médico	AC_AM	50 bytes	50 Kbps

A identificação antecipada de eventos críticos na saúde de pacientes foi realizada no R por meio do conjunto de indicadores estatísticos [Dakos et al. 2012]. A aplicação eficiente desses indicadores requer séries temporais com dados que possuem pelo menos um evento crítico. Por isso, e devido a escassa aplicação dos indicadores em dados empíricos, empregamos traços reais de frequência respiratória de um paciente masculino de 70 anos com edema pulmonar, ilustrados na Figura 3 (a). Esses dados encontram-se

disponíveis na base de dados MIMIC II do *PhysioNet Bank*[3]. Esse paciente foi monitorado durante 11 horas e 11 minutos na UTI de um hospital, onde em torno do minuto 15, o paciente sofreu uma insuficiência respiratória que durou em torno de 14 minutos (destacado pela caixa vermelha na figura). Como o paciente estava no hospital usando aparelhos respiratórios conseguiu voltar à estabilidade vital [Moody and Mark 1996].



(a) Série Temporal Completa (b) Série Temporal de 14 minutos

Figura 3. Traços Reais de Frequência Respiratória

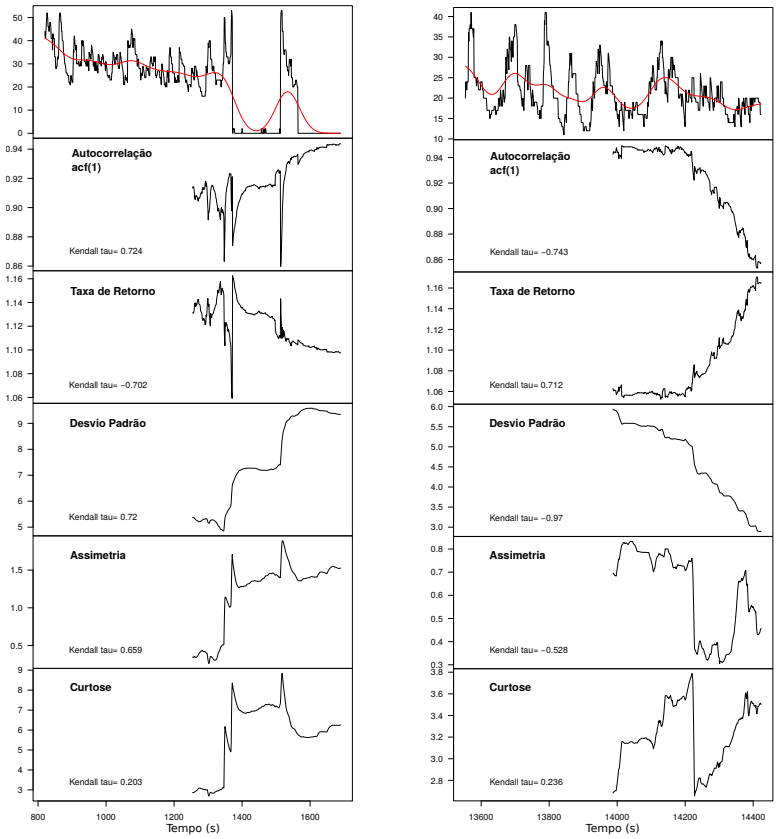
Como os indicadores apontam a tendência de um evento crítico, com o intuito de prevê-lo, quebramos a série temporal completa em séries temporais menores com janelas de tempo de 14 minutos cada. Similar aos experimentos realizados na literatura [Dakos et al. 2012], as séries temporais foram assim dimensionadas para englobar o evento crítico alvo que teve duração de aproximadamente 14 minutos e com isso, identificá-lo na série temporal que o precede (Figura 3 (b)). Para isso, aplicamos os indicadores em cada uma dessas séries e assim, esperamos que o conjunto de comportamento dos indicadores, para identificação do evento alvo, ocorra nesta série temporal que antecipa o evento. Em situações que não se sabe a duração do evento crítico, deve se utilizar janelas de tempo pequenas com poucos minutos, visto que os indicadores apresentam o comportamento esperado apenas próximo às mudanças abruptas. Quando identificado o evento, alertas médicos são gerados e priorizados no NS-3 com base no resultado do coeficiente de correlação *Kendall tau* de cada indicador. Esse coeficiente mede a força da tendência da curva dos indicadores, onde quanto maior a força, maior a probabilidade da ocorrência de um evento crítico. Em função da identificação do alerta ter sido feita no R, enquanto a priorização, transmissão e o cenário de simulação no NS-3, fizemos algumas suposições: (i) os sinais vitais são coletados pelos sensores de uma WBAN e enviados ao coordenador da WBAN (dispositivo móvel com aplicações médicas); (ii) o AP tem recursos computacionais suficiente para retransmissão dos dados recebidos ao servidor do hospital.

Para avaliar o desempenho dos indicadores estatísticos sobre sinais vitais reais, consideramos os indicadores de autocorrelação, taxa de retorno, assimetria, curtose e desvio padrão. No NS-3, foram medidos os valores médios das métricas de atraso médio e taxa de perda de pacotes. Os resultados, quando não mencionados de outra forma, são intervalos de confiança, com 95% de nível de confiança. O atraso médio consiste da soma da latência de todos os pacotes recebidos dividido pelo número total de pacotes recebidos. A taxa de perda é a diferença do número total de pacotes transmitidos pelo número total de pacotes recebidos. Cada uma dessas medidas estatísticas foi calculada para cada fluxo de tráfego e cenário considerado na simulação.

[3]PhysioBank, https://physionet.org/physiobank/. Último acesso em Dez/2016.

4.2. Resultados

A Figura 4 apresenta os resultados obtidos na aplicação do conjunto de indicadores estatísticos na identificação antecipada de eventos críticos na saúde de pacientes sobre sinais vitais de frequência respiratória. Os dados ilustrados no primeiro gráfico da Figura 4(a) apresentam uma transição crítica em direção ao evento de insuficiência respiratória. Há uma alteração de estado durante a transição, onde os valores alteraram de 50 para 0 na frequência respiratória. Os dois últimos minutos da série temporal compreendem os estágios iniciais do evento crítico, que foram necessários para estimação dos indicadores. Enquanto os dados ilustrados no primeiro gráfico da Figura 4(b) consistem de sinais respiratórios normais variando entre 10 e 40 os valores das observações, sem apresentar mudanças abruptas. Estimamos os indicadores em ambas as séries temporais a fim de comparar o conjunto de comportamento dos indicadores em situações críticas e normais.



(a) Durante transição crítica (b) Após evento crítico

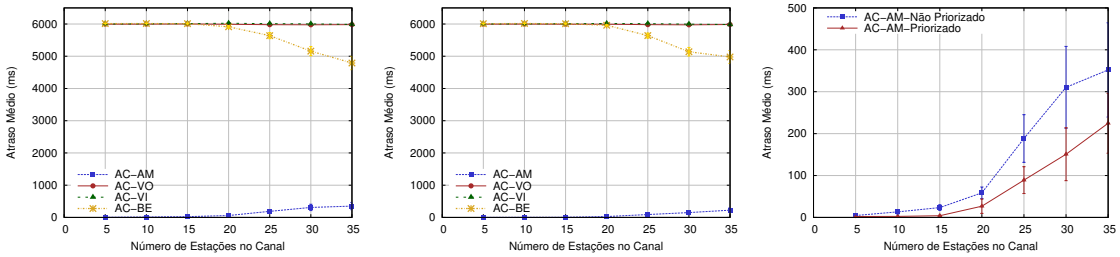
Figura 4. Indicadores Genéricos durante e após Transição Crítica

O comportamento dos indicadores nos gráficos da Figura 4(b) apresentam o inverso do esperado, pois os dados de respiração desta janela de tempo não possuem alterações abruptas, ou seja, os valores dos sinais vitais permaneceram em torno de valores estáveis. Por outro lado, na série temporal com transição crítica ilustrada na Figura 4(a), os resultados dos indicadores apresentam o comportamento genérico esperado para a identificação da iminência de eventos críticos. A autocorrelação aumentou consideravelmente, apontando forte relação entre as observações respiratórias com tendência de permanecer com valores em torno de 0, e ocorrer de fato o evento crítico. A queda na taxa de retorno confirma que o paciente sofreu perturbações próximo ao evento de insuficiência respiratória. Além disso, a variação entre 50 e 0 dos valores das observações

respiratórias, causou um aumento na variância dos dados medida pelo desvio padrão, o que aponta instabilidade no estado de saúde do paciente.

Além disso, a queda dos valores das observações justifica o aumento da assimetria positiva da distribuição dos dados. Isso indica que há uma alta concentração dos sinais respiratórios em torno de valores baixos, apontando assim a tendência do estado da doença permanecer em um estado crítico nas próximas observações. Também, a existência de valores extremos e raros nos dados observados (como valor 0) causa um aumento na curtose, indicando que o estado da doença não se encontra estável. Esse conjunto de comportamentos dos indicadores apresentando um aumento na autocorrelação, desvio padrão, assimetria e curtose, associado com uma queda na taxa de retorno aponta a possibilidade de ocorrer um evento emergencial na saúde do paciente. Como sabemos que ocorre de fato o evento de insuficiência respiratória na janela de tempo subsequente à janela utilizada para obtenção dos indicadores da Figura 4(a), e os resultados apresentam o comportamento visto na literatura, conseguimos mostrar que tais indicadores estatísticos podem indicar a iminência de eventos críticos em sinais vitais reais. Em nossos resultados, foi possível identificar o evento de insuficiência respiratória 12 min antes de acontecer o evento. Além disso, o uso dos indicadores para a identificação do evento não necessitou de parâmetros específicos do sinal vital empregado nas análises, permitindo assim seu uso genérico.

O sistema SANTE gera alertas médicos quando identificado eventos críticos, a fim de informar os profissionais da saúde. Os gráficos da Figura 5(a) e 5(b) apresentam o atraso médio observado na transmissão dos alertas médicos e dos demais tipos de tráfego considerados sobre os cenários com e sem o sistema proposto. Com pouca quantidade de dispositivos móveis (estações), ambos os cenários apresentam atrasos médios baixos para os alertas médicos. No entanto, conforme o número de dispositivos aumenta na rede, ficam evidente os ganhos que a nova abordagem de identificação de eventos e priorização de tráfego trás. Conforme a Figura 5(c), por exemplo, o cenário com 20 dispositivos e sem o uso do novo sistema apresenta atrasos, na média, de 58 ms. Por outro lado, nas mesmas condições, o uso do novo sistema reduz os atrasos para cerca de 26 ms. Em um cenário ainda mais denso, com 25 dispositivos, o novo sistema é capaz de apresentar atrasos inferiores a 89 ms, enquanto um sistema tradicional tem atrasos médios de 188 ms. Em especial, em cenários densos com 30 e 35 dispositivos, o cenário sem o uso do sistema proposto não consegue suportar a latência máxima desejada para aplicações médicas (125 ms) alcançando atrasos de 350 ms. Quando utilizamos o novo sistema, reduzimos esse atraso para cerca de 224 ms. Finalmente, a adoção do SANTE não apresenta impactos consideráveis nos demais tipos de tráfego.

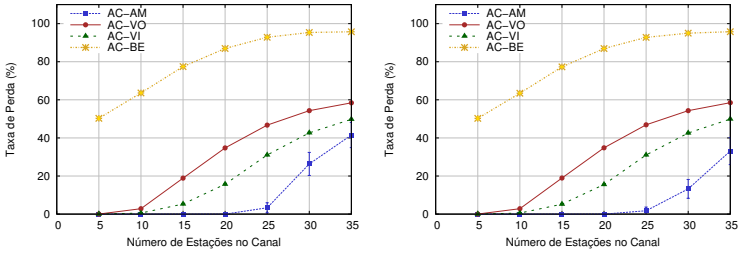


(a) Sem o SANTE (b) Com o SANTE (c) Atraso Alertas Médicos

Figura 5. Atraso Médio *versus* Densidade da Rede

A Figuras 6(a) e 6(b) apresentam a taxa de perda de pacotes dos fluxos de dados

sobre os dois cenários considerados. Em ambos os cenários, os alertas médicos foram entregues em sua totalidade considerando pouco número de dispositivos móveis no canal. Os demais tipos de tráfego considerados apresentam resultados semelhantes nos dois cenários. Porém, conforme a quantidade de dispositivos móveis aumentou, a taxa de perda aumentou para todos os tráfegos dada a saturação do canal. Entretanto, são visíveis os ganhos que o sistema proposto oferece para o tráfego dos alertas médicos. Em um cenário denso, com 30 dispositivos móveis e sem o uso do sistema SANTE, a taxa de perda foi em cerca de 26%. Em contrapartida, o mesmo número de dispositivos com o SANTE apresenta taxas de perda de 13%. Um cenário mais denso e sem o uso do SANTE apresenta cerca de 41% de perdas. Quando utilizamos o SANTE, nas mesmas condições, as perdas reduzem para 33%. Por fim, as perdas dos outros tráfegos não sofrem impacto.



(a) Sem o SANTE (b) Com o SANTE

Figura 6. Taxa de Perda *versus* Densidade da Rede

5. Conclusões

Neste artigo, nós apresentamos o SANTE (do inglês, **S***ystem for* **A***nticipated Identificatio***N** *and* **T***ransmission of M***E***dical Alerts on WBAN/WLAN*), o primeiro sistema para identificação antecipada e transmissão prioritária de alertas médicos no contexto híbrido WBAN/WLAN. O SANTE tem por objetivo predizer tendências no comportamento dos sinais vitais sobre a iminência de eventos críticos por meio de um conjunto genérico de indicadores estatísticos. Quando previsto algum evento crítico, o sistema gera alertas médicos e atribui aos mesmos o maior nível de prioridade de acesso ao meio da WLAN. Desse modo, o sistema oferece um tratamento adequado aos dados emergenciais e permite que os profissionais da saúde reajam em tempo hábil. De forma geral, os resultados de nossas simulações apontam para a viabilidade da aplicação dos indicadores estatísticos para predição de eventos críticos em sinais fisiológicos. De fato, o SANTE foi capaz de identificar o evento crítico avaliado e assim, reduzir em 39% o atraso nas transmissões de alertas médicos além de diminuir significativamente as perdas. Além disso, o SANTE não causa impacto sobre os outros tipos de tráfego, como voz e vídeo. Como trabalhos futuros, pretendemos reduzir ainda mais o atraso médio dos alertas, em cenários densos. Adicionalmente, aplicar os indicadores em outros tipos de sinais vitais, a fim de ajustá-los ainda mais e assim, reduzir a janela de dados críticos utilizada na estimação dos mesmos.

Referências

Association, I. S. et al. (2012). 802.15. 6-2012 IEEE standards for local and metropolitan area networks–part 15.6: Wireless body area networks.

Bhandari, S. and Moh, S. (2016). A priority-based adaptive mac protocol for wireless body area networks. *Sensors*, 16(3):401.

Bradai, N., Charfi, E., Fourati, L. C., and Kamoun, L. (2016). Priority consideration in inter-WBAN data scheduling and aggregation for monitoring systems. *Trans. Emerg. Telecommun. Technol.*, 27(4):589–600.

Bradai, N., Fourati, L. C., and Kamoun, L. (2015). Wban data scheduling and aggregation under wban/wlan healthcare network. *Ad Hoc Netw.*, 25:251–262.

Cavallari, R., Martelli, F., Rosini, R., Buratti, C., and Verdone, R. (2014). A survey on wireless body area networks: technologies and design challenges. *IEEE Commun. Surveys Tuts.*, 16(3):1635–1657.

Cisco (2006). Diffserv-the scalable end-to-end QoS model. *CiscoSystems*.

Dakos, V., Carpenter, S. R., Brock, W. A., Ellison, A. M., Guttal, V., Ives, A. R., Kéfi, S., Livina, V., Seekell, D. A., van Nes, E. H., and Scheffer, M. (2012). Methods for detecting early warnings of critical transitions in time series illustrated using simulated ecological data. *PloS one*, 7(7):e41010.

Divgi, G. and Chlebus, E. (2013). Characterization of user activity and traffic in a commercial nationwide WiFi hotspot network: global and individual metrics. *Wireless Netw.*, 19(7):1783–1805.

Gündoğdu, K. and Çalhan, A. (2016). An implementation of wireless body area networks for improving priority data transmission delay. *J. Med. Syst.*, 40(3):1–7.

IEEE (2012). IEEE standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements–part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Technical Report Std. 802.11*.

Kathuria, M. and Gambhir, S. (2016). A novel optimization model for efficient packet classification in wban. *International J. Energy, Inf. Commun.*, 7(4):1–10.

Kim, R. H. and Kim, J. G. (2015). Adaptive mac protocol for emergency data transmission in wireless body sensor networks. *International J. Softw. Eng. and Its Appl.*, 9(9):205–216.

Latré, B., Braem, B., Moerman, I., Blondia, C., and Demeester, P. (2011). A survey on wireless body area networks. *Wireless Netw.*, 17(1):1–18.

Misra, S. and Sarkar, S. (2015). Priority-based time-slot allocation in wireless body area networks during medical emergency situations: An evolutionary game-theoretic perspective. *IEEE J. Biomed. Health Inform.*, 19(2):541–548.

Moody, G. B. and Mark, R. G. (1996). A database to support development and evaluation of intelligent intensive care monitoring. In *Comput. in Cardiology*, pages 657–660. IEEE.

Movassaghi, S., Abolhasan, M., Lipman, J., Smith, D., and Jamalipour, A. (2014). Wireless body area networks: A survey. *IEEE Commun. Surveys Tuts.*, 16(3):1658–1686.

Rashwand, S. and Misic, J. V. (2015). Bridging between IEEE 802.15. 6 and IEEE 802.11 e for wireless healthcare networks. *Ad Hoc & Sensor Wireless Netw.*, 26(1-4):303–337.

Rikkert, M. G. O., Dakos, V., Buchman, T. G., de Boer, R., Glass, L., Cramer, A. O., Levin, S., van Nes, E., Sugihara, G., and Ferrari, M. D. (2016). Slowing down of recovery as generic risk marker for acute severity transitions in chronic diseases. *Critical Care Med.*, 44(3):601–606.

Scheffer, M., Bascompte, J., Brock, W. A., Brovkin, V., Carpenter, S. R., Dakos, V., Held, H., Van Nes, E. H., Rietkerk, M., and Sugihara, G. (2009). Early-warning signals for critical transitions. *Nature*, 461(7260):53–59.

Silva, R., Achir, N., Santos, A., and Nogueira, M. (2016). Avoiding collisions by time slot reduction supporting voice and video in 802.11 networks. In *IEEE GLOBECOM*, pages 1–6.

van de Leemput, I. A., Wichers, M., Cramer, A. O., Borsboom, D., Tuerlinckx, F., Kuppens, P., van Nes, E. H., Viechtbauer, W., Giltay, E. J., Aggen, S. H., et al. (2014). Critical slowing down as early warning for the onset and termination of depression. *Proc. National Academy Sci.*, 111(1):87–92.

Reduzindo a latência de comunicação em múltiplos saltos dos mecanismos de *duty cycle* assíncrono baseados em *schedule* através de sincronização de baixa resolução

André R. C. Saraiva[1], **Diego Passos**[1], **Ricardo C. Carrano** [2] **e Célio V. N. Albuquerque** [1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
[2]Departamento de Engenharia de Telecomunicações – Universidade Federal Fluminense (UFF)
Av. Gal. Milton Tavares de Souza, s/nº – 24.210-346 – São Domingos – Niterói – RJ – Brasil

{andresaraiva,dpassos,celio}@ic.uff.br

carrano@midiacom.uff.br

Abstract. *In Wireless Sensor Networks, nodes typically employ batteries that cannot be recharged or replaced. Hence, optimizing energy consumption is a major concern. Among the several solutions already proposed, schedule-based asynchronous duty cycle methods are the simplest because they do not require mechanisms, protocols or specific hardware for clock synchronization between nodes. These solutions, however, result in high latency for multi-hop communication. In this work, we show how existing asynchronous mechanisms can benefit from a low level of synchronism, with resolution of slots. Assuming this possibility, we show, using numerical simulations, that the use of specific offsets between clocks of neighboring nodes according to their distances to the sink node drastically reduces latency.*

Resumo. *Nas Redes de Sensores Sem Fio, nós tipicamente operam por baterias não recarregáveis ou substituíveis. Logo, a otimização do consumo energético é uma das principais preocupações. Dentre as várias soluções já propostas, os métodos de duty cycle assíncronos baseados em schedule se mostram os mais simples por não demandarem mecanismos, protocolos ou hardwares específicos para sincronização de relógio entre os nós. Estes métodos, no entanto, resultam em alta latência para comunicação de múltiplos saltos. Neste trabalho, demonstramos como os mecanismos assíncronos já existentes podem se beneficiar de um baixo nível de sincronismo, em resolução de slots. Assumindo esta possibilidade, mostramos com simulações numéricas que o uso de offsets específicos entre os relógios de nós vizinhos, de acordo com suas distâncias ao nó sorvedouro, reduz drasticamente a latência.*

1. Introdução

Em um mundo globalizado, com ênfase na sustentabilidade, a eficiência energética dos equipamentos eletrônicos é de suma importância. Em particular, as redes sem fio de múltiplos saltos, como as Redes de Sensores Sem Fio (RSSF), apresentam severas limitações energéticas por comumente dependerem de fontes de energia portáteis. Além das limitações energéticas, a troca da fonte de energia de nós sensores, na maioria das vezes, não é trivial ou mesmo possível, fazendo com que o gerenciamento de energia seja vital para operação adequada da RSSF. O *duty cycling* da interface de rádio,

que é um dos componentes responsáveis pela drenagem de energia, se torna necessário [Anastasi et al. 2009, Bachir et al. 2010], alternando períodos de atividade e inatividade.

Entretanto, fazer com que todos os elementos de uma RSSF mantenham de forma coordenada seus períodos de atividade e inatividade a fim de garantir que um nó sempre encontre um nó vizinho para transmitir seus dados com sucesso requer mecanismos síncronos, com um relógio de tempo comum compartilhado por todos os nós, ou assíncronos, onde um relógio comum não se faz necessário.

Mecanismos síncronos comumente requerem *hardware* adicional, geram um maior tráfego de controle e resultam em maiores custos [Carrano et al. 2014a]. Em contrapartida, mecanismos assíncronos não demandam *hardware* especializado ou a adição de grande volume de tráfego de controle, se mostrando, portanto, uma alternativa interessante no contexto de redes de sensores. Porém, segundo [Ye et al. 2002, Lu et al. 2004], os mecanismos assíncronos apresentam uma latência excessiva ao longo de caminhos múltiplos saltos.

Mesmo assim, diversas propostas assíncronas baseadas em *schedules* podem ser encontradas na literatura, como *Block Design* [Zheng et al. 2003], *Grid* [Jiang et al. 2005], *Torus* [Tseng et al. 2003] e *Disco* [Dutta and Culler 2008]. No entanto, estas propostas não abordam a questão da elevada latência fim-a-fim.

O problema de *sleep waiting* e o problema de *data forwarding interruption* [Lu et al. 2004] são exemplos de fenômenos que contribuem para o aumento da latência, uma vez que um transmissor deverá aguardar certo tempo até que ele e o receptor pretendido fiquem, ambos, com seus rádios ligados simultaneamente (o que constitui uma oportunidade de encontro). Estes fenômenos fazem com que o tempo de descoberta de vizinho (ou NDT, do inglês *Neighbor Discovery Time*) seja elevado.

Embora não seja o escopo deste trabalho apresentar um protocolo de sincronização, neste artigo, é estudada a possibilidade de introduzir um certo grau de sincronismo de baixa resolução em mecanismos assíncronos, sem que haja a necessidade de *hardware* adicional ou aumento significativo de tráfego de controle, mas proporcionando uma redução representativa do NDT. Para tanto, considera-se a possibilidade de existência de um protocolo de sincronização dos nós em resolução de *slots* que permita que nós vizinhos sigam escalonamentos com *offsets* específicos (ao invés dos *offsets* aleatórios resultantes da natural dessincronização dos relógios dos nós). Através do uso de simulações numéricas, este trabalho demonstra que a utilização de certos valores específicos de *offset* reduz consideravelmente a latência da comunicação em múltiplos saltos.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta uma revisão bibliográfica sobre alguns mecanismos de *duty cycle* síncronos e assíncronos. A Seção 3 mostra como o uso de uma sincronização de baixa resolução pode reduzir a latência da comunicação em múltiplos saltos. A Seção 4 avalia os benefícios da sincronização proposta através de simulações numéricas. Por fim, a Seção 5 apresenta considerações finais e ideias para trabalhos futuros.

2. Revisão Bibliográfica

Mecanismos de *duty cycling* de rádio têm como objetivo reduzir o tempo de escuta ociosa (*i.e.*, intervalos em que o rádio se encontra ligado sem que haja transmissões ou recepções

de quadros de interesse) dos nós de uma rede sem fio, reduzindo assim o consumo de energia e aumentando a vida útil da rede. Na literatura da área, define-se o *duty cycle* de um nó como o percentual de tempo que este mantém seu rádio ligado [Carrano et al. 2014b]. Considerando os requisitos e capacidades atuais dos nós sensores, *duty cycles* inferiores a 1% são desejáveis [Carrano et al. 2014b]. Em contrapartida, a realização de *duty cycling* do rádio não é trivial, já que algum nível de coordenação é necessário para garantir que quaisquer dois nós vizinhos ainda tenham oportunidades de comunicação (*i.e.*, que exista algum período suficientemente longo em que os nós tenham seus rádios simultaneamente ligados). Ademais, a redução do *duty cycle* dos nós de uma rede geralmente é associada a um aumento da latência de comunicação. Por estes motivos, a comunidade acadêmica tem realizado esforços para obter soluções viáveis com *duty cycles* baixos e, idealmente, baixo *overhead* de controle [Zheng et al. 2003].

2.1. Principais Mecanismos Síncronos

Como forma de sincronizar o período de atividade dos rádios, reduzindo a latência, a escuta ociosa e a perda de pacotes, os esquemas síncronos introduzem um relógio comum entre todos os nós da rede. O estabelecimento deste relógio comum demanda a utilização permanente de protocolos de sincronização de relógio ou a adição de *hardware* especializado para manter a sincronização com o grau de precisão necessário. Estes mecanismos podem ser classificados em duas famílias: os esquemas estritamente síncronos ou baseados em *Rendezvous* [Sivrikaya and Yener 2004] e os esquemas baseados em escalonamento [Anastasi et al. 2009].

Os esquemas estritamente síncronos são os de mais fácil compreensão, uma vez que todos os nós deverão ter seus rádios ligados ou desligados ao mesmo tempo. No entanto, em múltiplos saltos este esquema torna-se complexo, já que erros de sincronização entre os nós são comuns e tendem a se acumular ao longo dos saltos, dificultando a sincronização da rede como um todo e, muitas vezes, demandando elementos adicionais de *hardware* e certa quantidade de mensagens de controle [Sivrikaya and Yener 2004].

Exemplos de esquemas estritamente síncronos encontrados na literatura são o RT-Link [Rowe et al. 2008], que acrescenta um GPS para sincronização, e TRAMA [Rajendran et al. 2006], que tem como princípio a eliminação das colisões e a suspensão dos nós que não participam da comunicação naquele instante de tempo. Ambos utilizam o protocolo de acesso múltiplo TDMA, do inglês *Time Division Medium Access*, evitando colisões entre pacotes, transmissões desnecessárias e escutas ociosas, e consequentemente desperdício de bateria, características estas que estão entre as principais vantagens destes esquemas.

Outras propostas são os esquemas baseados em escalonamento, que visam resolver o problema de interrupção de encaminhamento de dados (do inglês *Data Forwarding Interruption Problem*) – que segundo [Lu et al. 2004] ocorre quando o percurso de um pacote pela rede é interrompido devido ao próximo nó no caminho tornar seu rádio inativo – e reduzir o tempo de espera por inatividade, uma vez que se pretende manter um sincronismo para que um nó não fique ativo prematuramente, aguardando até que seu nó vizinho se ative. Estas soluções utilizam uma topologia em árvore, cuja raiz é o nó sorvedouro e os demais nós são ativados em instantes de tempo determinados de acordo com sua profundidade na árvore.

Diversos esquemas baseados em escalonamento foram propostos na literatura, entre eles o SPEED-MAC [Choi et al. 2010], que introduziu um período de sinalização para notificação de eventos e preparo dos nós para transmitirem os dados recebidos de comunicações prévias, e o CUPID [Kruger et al. 2010], que propõe um esquema bidirecional para o tráfego na árvore, apontando não só a existência de fluxo dos sensores para o nó sorvedouro, mas também um fluxo de configuração do nó sorvedouro para os sensores.

Uma dificuldade dos esquemas baseados em escalonamento é que, mesmo sendo menos restritivos em relação ao nível de sincronização dos nós como um todo (basta garantir um nível de sincronização entre nós vizinhos), ainda é necessária uma sincronização de relógio com alta resolução, o que vem associado à adição de *hardware* especializado e/ou alto *overhead* de sincronização. Além disso, no caso da entrada de novos nós na rede, estes deverão estar previamente sincronizados com o restante da rede (solução mais complexa) ou deverão entrar em um estado inicial de sincronização no qual seus rádios permanecerão constantemente ligados, induzindo um alto consumo energético até a sincronização.

2.2. Principais Mecanismos Assíncronos

Diferentemente dos mecanismos síncronos, os mecanismo assíncronos são menos complexos, em termos de adição de *hardware*, mensagens de controle e custo. Por esta razão, a tendência ao assincronismo tem ganhado espaço na comunidade científica, em especial aos mecanismos baseados em escalonamento. Porém, problemas com a latência e escuta ociosa ainda continuam como uma preocupação.

O mecanismo de amostragem de preâmbulo [Polastre et al. 2004] faz com que cada nó fique inativo de forma assíncrona, ativando-se periodicamente para escutar o meio. Se, ao escutar o meio, o nó detecta um preâmbulo, este permanece com seu rádio ligado para receber o quadro completo.

Já [Sun et al. 2008] classifica alguns mecanismos baseados na iniciativa do receptor. Nestes métodos, o transmissor aguarda um sinal do receptor, indicando que este está apto a receber o dado. Neste caso, a demanda energética é mais alta para o transmissor, que deve aguardar com seu rádio ligado até que o receptor acorde.

O *On-demand wakeup* [Schurgers et al. 2002] assume a existência de uma segunda interface de rádio de baixo consumo que permanece sempre ligada através da qual o nó receptor pode ser avisado de que deve ligar sua interface principal para a transmissão de dados.

O *duty cycling* aleatório [Paruchuri et al. 2004] adequa-se a RSSF densas. Os nós ligam e desligam seus rádios aleatoriamente, sem coordenação prévia. Quando um nó deseja transmitir um quadro, ele o faz independentemente de quais outros nós estejam ativos naquele momento. Se a rede for densa o suficiente, há alta probabilidade de que pelo menos um receptor receba o quadro transmitido. O método, portanto, assume que as transmissões na camada de enlace não são endereçadas a um próximo salto específico.

Finalmente, existem os mecanismos baseados em *schedules*, tais como *Block Design, Grid, Torus* e *Disco*, que são caracterizados por dividir o tempo em ciclos, por sua vez subdivididos em *slots* ativos e inativos. O grande diferencial destas propostas é que os padrões de *slots* ativos e inativos que compõem um ciclo são especialmente projetados

com o intuito de garantir ao menos uma sobreposição de *slots* ativos por ciclo entre quaisquer dois nós, independentemente do *offset* relativo entre seus respectivos relógios locais, uma característica conhecida na literatura como fechamento para rotação (do inglês, *rotation closure*) [Carrano et al. 2014a]. Ao contrário dos demais mecanismos assíncronos discutidos até aqui, estes mecanismos não assumem a existência de um segundo rádio de baixo consumo, nem requerem a escuta ao meio por um sinal ou uma grande densidade de nós.

Um *Block Design* [Zheng et al. 2003] $\{v, k, \lambda\}$ é uma entidade matemática composta por um conjunto V de v elementos inteiros, juntamente com uma família de v subconjuntos de V (chamados blocos), cada um com exatamente k elementos, tal que a interseção entre quaisquer dois blocos diferentes tenha exatamente λ elementos. No caso particular em que λ=1, os *Block Designs* recebem o nome de plano projetivo.

Um bloco de um *Block Design* pode ser mapeado para um padrão de *slots* ativos e inativos da seguinte forma. Cada *slot* do padrão corresponde a um elemento em V. Para cada elemento em V, se este pertence ao bloco, então ativa-se o *slot* correspondente no padrão. Caso contrário, o *slot* correspondente será inativo. A Figura 1 apresenta um exemplo com dois nós seguindo padrões baseados em blocos diferentes do *Block Design* $\{7,3,1\}$ onde temos V=$\{1,2,3,4,5,6,7\}$, k = 3, resultando nos seguintes blocos: $\{$[1,2,4], [2,3,5], [3,4,6], [4,5,7], [5,6,1], [6,7,2], [7,1,3]$\}$.

Note que os padrões gerados por cada bloco são simplesmente rotações dos padrões de outros blocos do mesmo *Block Design*. Logo, se dois nós de uma rede seguem um padrão resultante de um mesmo bloco, porém possuem seus relógios dessincronizados, isto é equivalente a cada nó utilizar um bloco diferente do mesmo *Block Design* sob uma base de tempo global. Com isso, os *Block Designs* garantem a existência de ao menos λ *slots* ativos coincidentes por ciclo, independentemente do *offset* relativo entre os relógios dos nós.

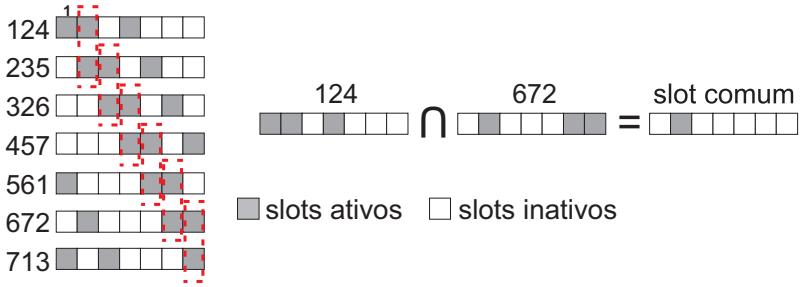


Figura 1. Mapeamento de cada bloco do *Block Design* $\{7, 3, 1\}$ para um padrão de *slots* ativos e inativos. À direita, ilustra-se o fechamento para rotação, com a interseção entre padrões de dois blocos distintos resultando em exatamente um *slot* em comum.

O *duty cycle* para um *Block Design* é dado por $\frac{k}{v}$. Em particular, é possível demonstrar que planos projetivos constituem os padrões de menor *duty cycle* com fechamento para rotação para um dado tamanho de ciclo [Maekawa 1985]. No entanto, o número de *Block Designs* conhecidos é limitado. Em particular, não são conhecidos planos projetivos com *duty cycles* menores que 1,03% [Link et al. 2011].

Em [Carrano et al. 2014a], é derivada uma expressão que aproxima o NDT (*i.e.*, o

tempo esperado até a comunicação efetiva entre dois nós vizinhos) para um *Block Design* $\{v, k, \lambda\}$ em função de p, a probabilidade de entrega de quadros do enlace:

$$E[NDT] = \frac{v+1}{p(\lambda+1)} - \frac{(v+1)(1-p)^{\lambda} - (\lambda+1)}{(\lambda+1)[(1-p)^{\lambda} - 1]} \quad (1)$$

O *Grid* [Jiang et al. 2005] é um dos mecanismos baseados em sistemas de quórum. O *Grid* define uma lei de formação de padrões de *slots* ativos e inativos que, assim como os *Block Designs*, garante o fechamento para rotação. A Figura 2 mostra um exemplo de um padrão gerado por um *Grid* 5x5. Cada nó seleciona uma linha e uma coluna em uma matriz n x n. Todos os *slots* da linha e da coluna selecionadas serão ativos, enquanto os demais *slots* serão inativos. O padrão resultante é obtido através da linearização da matriz, *i.e.*, o padrão é formado pela sequência de *slots* da primeira linha, seguidos dos *slots* da segunda e assim sucessivamente, resultando em um ciclo de n^2 *slots*. Mesmo o método funcionando com a escolha de linhas e colunas diferentes por cada nó, na prática comumente todos os nós escolhem as mesmas linha e coluna, resultando no mesmo padrão linearizado, embora, possivelmente, em *offsets* diferentes por conta da falta de sincronização, conforme ilustrado na Figura 2 (A e B seguem o mesmo padrão, mas com um *offset* relativo de um *slot*).

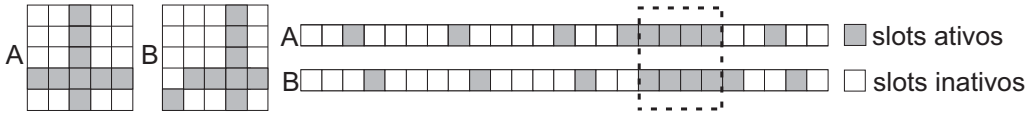


Figura 2. Dois nós A e B operando de acordo com o padrão gerado por um *Grid* de ordem 5, com um *offset* relativo de 1 *slot*. O padrão garante ao menos, dois *slots* ativos comuns em cada ciclo, independentemente do *offset* entre os nós.

O *duty cycle* no *Grid* é dado por $\frac{2n-1}{n^2}$, onde n é a ordem da matriz. A Equação 2, extraída de [Carrano et al. 2014a], fornece uma estimativa do NDT para um *Grid* de ordem n na comunicação por um enlace com probabilidade de entrega de quadros p:

$$E[NDT] = \frac{(3-p)n^2}{6p} \quad (2)$$

O *Torus* [Jiang et al. 2005], outro método da família dos sistemas de quórum, é uma proposta de melhoria do *Grid* em termos de *duty cycle*. Embora sua lei de formação de padrões seja similar à do *Grid*, ao invés de se ativarem todos os *slots* da linha selecionada, são ativados apenas metade mais um destes, resultando em um padrão de comprimento n^2, mas com menos *slots* ativos em comparação ao *Grid*. A Figura 3 exemplifica essa lei de formação, mostrando os *slots* ativos coincidentes para dois nós com *offset* relativo de um *slot*.

O *duty cycle* de um padrão resultante de um *Torus* de ordem n é dado por $\frac{3}{2n}$, para n par, ou $\frac{3n-1}{2n^2}$, para valores de n ímpares. Já o NDT estimado, segundo [Carrano et al. 2014a], é dado por:

$$E[NDT] = \frac{(2-p)n^2}{2p} \tag{3}$$

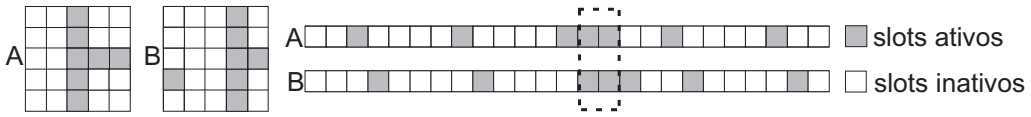


Figura 3. Dois nós A e B operando de acordo com o padrão gerado por um *Torus* de ordem 5, com um *offset* relativo de 1 *slot*. O padrão garante pelo menos um *slot* ativo em comum, independentemente do *offset* relativo entre os nós.

O *Disco* [Dutta and Culler 2008], um mecanismo baseado em números primos, garante que se os *slots* ativos de dois nós A e B são múltiplos de q_1 ou de q_2 (os números primos que são parâmetros do método), sempre ocorrerá uma sobreposição de *slots* ativos, independentemente dos *offsets* relativos de seus relógios locais.

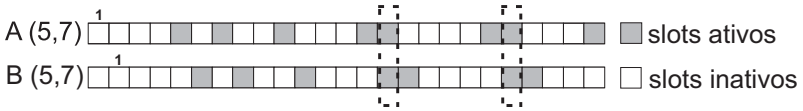


Figura 4. Dois nós A e B operando com o padrão gerado pelo mesmo Disco ($\{5, 7\}$) e com um *offset* relativo de 1 *slot* . O padrão garante a existência de *slots* ativos comuns independentemente do *offset* relativo entre os nós.

Segundo [Carrano et al. 2014a], o NDT resultante da utilização do *Disco* com os parâmetros q_1 e q_2 é aproximadamente dado por:

$$E[NDT] = \frac{q_1 q_2 (p^2 - 3p + 3)}{3p(2-p)} \tag{4}$$

3. Sincronização de *Slots* em Múltiplos Saltos

Como citado anteriormente, não é o escopo deste trabalho apresentar um protocolo de sincronização, mas analisar a possibilidade de ganho de desempenho na introdução de certo grau de sincronismo de baixa resolução em mecanismos assíncronos. A proposta se baseia na possibilidade de escalonar o uso do rádio de acordo com a distância de cada nó ao sorvedouro (em número de saltos) e não aleatoriamente, com base no momento em que o mesmo é ligado. Isso pode ser feito, por exemplo, através de adição da informação do *slot* de tempo atual de um nó quando este transmite seus pacotes de controle relativos ao próprio protocolo de roteamento. Ao receber um destes pacotes de um dado vizinho, o nó verificaria se este vizinho é, de acordo com as informações atuais de roteamento, seu próximo salto em direção ao sorvedouro. Em caso afirmativo, o nó alteraria seu *slot* de tempo atual para o *slot* atual do vizinho somado, possivelmente, de algum *offset* configurável. O *offset* irá estabelecer o deslocamento entre o padrão no qual um nó deverá operar em relação aos seus vizinhos. A distância em que cada nó se encontra do sorvedouro ou a própria árvore de menores caminhos entre os sensores e o sorvedouro são tipicamente obtidas dos protocolos de roteamento.

Neste trabalho, assumimos que um *slot* seja longo o suficiente para a transmissão de um quadro (incluindo a transmissão de um eventual *ACK* da camada de enlace, bem como outros tempos associados ao procedimento de acesso ao meio). Esta suposição é razoável, já que quanto menores os *slots*, menor é o comprimento de um ciclo do padrão de *duty cycle* o que, em uma escala de tempo absoluta, reduz a latência de comunicação. Uma consequência desta hipótese é que a perfeita sincronização dos padrões dos nós ao longo de um caminho de múltiplos saltos (*i.e.*, o uso de um *offset* = 0) não é uma boa escolha porque, ao receber um pacote para encaminhamento, um nó intermediário terá que esperar pelo próximo *slot* ativo em seu padrão, o que pode demorar a ocorrer. Na Figura 5, do lado esquerdo é mostrado um exemplo de um caminho de múltiplos saltos no qual todos os nós têm seus padrões de *duty cycle* perfeitamente alinhados (*i.e.*, *offset* = 0). Se o nó C estiver no *slot* quatro no momento da sua transmissão bem sucedida para o nó B, B precisará aguardar vários *slots* até que exista uma oportunidade de transmissão para A (*i.e.*, para que ambos estejam simultaneamente com seus rádios ligados). Por outro lado, se a transmissão bem sucedida de C para B se der no primeiro *slot* do padrão, B poderá realizar uma tentativa de encaminhamento do pacote logo no *slot* subsequente, já que este será um *slot* ativo para ambos B e C. Mesmo neste caso, no entanto, se existisse um terceiro salto neste caminho, (*e.g.*, um nó entre A e o sorvedouro), o nó A necessariamente teria que aguardar certa quantidade de *slots* para obter uma nova oportunidade de transmissão. Por fim, é importante notar que o *Block Design* $\{7, 3, 1\}$ possui quase 50% dos *slots* ativos, o que reduz o número de *slots* até a próxima oportunidade de transmissão. Se o padrão usado fosse o $\{9507, 98, 1\}$, com aproximadamente 1% dos *slots* ativos (um *duty cycle* muito mais adequado às aplicações atuais), a possibilidade de que um nó intermediário receba um pacote e possua uma oportunidade de transmissão logo no *slot* seguinte se torna bem mais baixa.

Ainda na Figura 5, do lado direito, é apresentado um esquema de escalonamento alternativo, no qual, ao invés de todos os nós operarem sob padrões perfeitamente alinhados, um *offset* de 1 *slot* é aplicado entre os padrões de nós vizinhos. O nó C, o mais distante do sorvedouro, ativa seus *slots* 1, 2 e 4 (de acordo com o *Block Design*$\{7, 3, 1\}$). Já o nó B, próximo salto de C em direção ao sorvedouro, tem seu padrão atrasado em um *slot* em relação a C, resultando no padrão $[1 + 1, 2 + 1, 4 + 1] = [2, 3, 5]$. Da mesma forma, o nó A precisa de um *offset* relativo a B de um *slot*, resultando no padrão $[2 + 1, 3 + 1, 5 + 1] = [3, 4, 6]$.

Este escalonamento garante que, quando um nó intermediário recebe um pacote para encaminhamento, este terá uma oportunidade de realizá-lo logo no *slot* subsequente. Se os enlaces da rede possuem alta probabilidade de entrega de quadros, este esquema mitiga o problema da interrupção do encaminhamento, reduzindo a latência de comunicação fim-a-fim.

Embora a Figura 5 ilustre o uso do *offset* = 1 para um padrão específico, a mesma análise pode ser realizada para qualquer padrão de *duty cycle* com fechamento para rotação. Qualquer padrão que apresente fechamento para rotação necessariamente possui dois *slots* consecutivos ativos (do contrário, dois nós com *offset* relativo igual a 1 nunca teriam *slots* ativos coincidentes). Logo, através da aplicação do *offset* = 1 entre vizinhos, sempre que um nó intermediário B recebe um pacote para encaminhamento, seu próximo *slot* será, necessariamente ativo. Como o próximo nó no caminho, digamos A, por sua

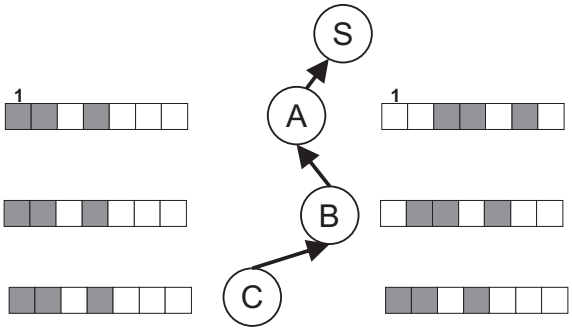


Figura 5. Exemplo de caminho de múltiplos saltos no qual cada nó opera segundo um padrão de *duty cycle* baseado no *Block Design* $\{7, 3, 1\}$. O nó S é o sorvedouro da rede. São ilustradas duas possíveis sincronizações: os padrões à esquerda possuem *offset* = 0, enquanto os padrões à direita têm um *offset* relativo de 1 *slot* entre vizinhos.

vez, também opera com um *offset* = 1 em relação a B, este também terá seu *slot* ativo. Este argumento pode ser facilmente estendido para um número arbitrário de saltos.

Em um caso extremo, se todos os enlaces do caminho têm probabilidade de entrega de quadros igual a 1, a latência fim-a-fim é dada pelo NDT do primeiro salto, mais um *slot* para cada salto subsequente. Em comparação, no caso de *offsets* aleatórios (*i.e.*, de uma rede sem qualquer sincronismo), a latência fim-a-fim se aproxima do produto do NDT pelo número de saltos do caminho.

Na prática, enlaces sem fio são sempre susceptíveis a algum nível de perda, tornando a expectativa de enlaces com a probabilidade de entrega de quadros igual a 1 irreal. Entretanto, como será mostrado nos resultados da nossa avaliação, mesmo para valores realísticos de probabilidade de entrega de bons enlaces sem fio, o uso do *offset* = 1 resulta em grande redução na latência fim-a-fim.

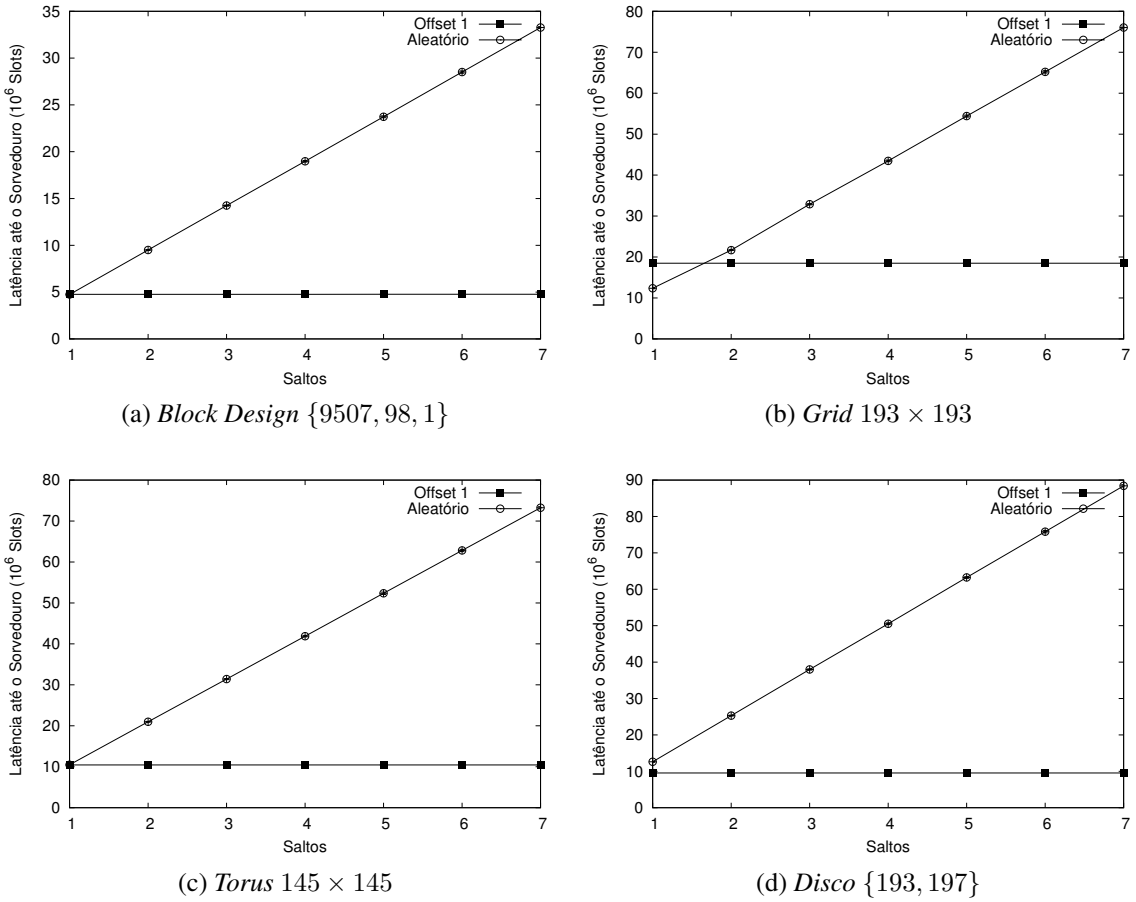
Por fim, é importante destacar as diferenças entre a presente proposta e mecanismos síncronos baseados em escalonamento, como o SPEED-MAC e o CUPID. Assim como estes dois mecanismos, nossa proposta também associa o escalonamento dos nós a suas distâncias em relação ao sorvedouro. No entanto, ao contrário do que ocorre nos mecanismos síncronos, o uso de esquemas assíncronos, como *Block Design*, *Grid*, *Torus* e *Disco*, simplifica a entrada de novos nós na rede, sem a necessidade de um sincronismo de relógio prévio ou de uma fase preliminar em que os novos nós fiquem com seus rádios permanentemente ligados. Um novo nó, ao entrar na rede, pode simplesmente seguir o padrão de *duty cycle* assíncrono, economizando energia e, ainda assim, com a garantia de que conseguirá se comunicar com seus vizinhos. Após algum tempo, através da troca de mensagens de controle com seus vizinhos, o nó realizará sua sincronização com os demais membros da rede, otimizando seu desempenho em relação a latência fim-a-fim.

4. Avaliação

Para avaliar a análise teórica apresentada na seção anterior, realizamos simulações numéricas comparando o funcionamento dos mecanismos assíncronos baseados em *schedules* com *offsets* aleatórios, resultantes da falta de sincronização de relógio entre os nós da rede, e o *offset* = 1 entre os vizinhos. Em mecanismos assíncronos baseados em *sche-*

dules, tais comparações se dão tradicionalmente em termos de métricas como consumo de energia e latência [Kandhalu et al. 2010, Link et al. 2011].

Foram realizadas simulações considerando comunicações de 1 a 7 saltos e diversos valores de probabilidade de entrega de quadros (variando de 0,05 a 1, com incrementos de 0,05). Para cada conjunto de parâmetros, as simulações foram repetidas 20000 vezes, permitindo a estimativa do NDT médio e o cálculo do intervalo de confiança de 95%. Embora nos resultados que se seguem os intervalos de confiança tenham sido plotados para todos os pontos de todos os gráficos, seus valores foram percentualmente muito baixos, dificultando sua visualização nas figuras. Foram comparados o *Block Design* $\{9507, 98, 1\}$, *Grid* 193x193, *Torus* 145x145 e *Disco* $\{193, 197\}$, todos resultando em *duty cycle* aproximado de 1,03%. O simulador utilizado foi o mesmo em [Carrano et al. 2013], estendido com a capacidade de simular comunicações em múltiplos saltos.



(a) *Block Design* $\{9507, 98, 1\}$ (b) *Grid* 193×193

(c) *Torus* 145×145 (d) *Disco* $\{193, 197\}$

Figura 6. Comparação da latência de comunicação obtida com *offset* aleatório e igual a 1 por padrões baseados em *Block Design*, *Grid*, *Torus* e *Disco* variando-se o número de saltos do caminho com p=1.

A Figura 6 mostra, para cada mecanismo de *duty cycle*, a latência de comunicação fim-a-fim como uma função do número de saltos do caminho para padrões com *duty cycles* de aproximadamente 1,03% e $p = 1$. Os gráficos mostram que o uso do *offset* = 1 resultou em menores latências para todas as comunicações de pelo menos dois saltos

em todos os quatro mecanismos avaliados. Tanto para o *offset* aleatório, quanto para o *offset* = 1, a latência fim-a-fim aumenta linearmente com o aumento do número de saltos. No entanto, a taxa de aumento é bem mais baixa para o *offset* = 1 (para este cenário, um aumento de apenas um *slot* por salto, conforme explicado na Seção 3). Para comunicações de um único salto, o *offset* = 1 resulta em latência superior em alguns dos mecanismos, já que embutido no caso aleatório está o *offset* = 0, que garante um número maior de oportunidades de encontro por ciclo. No entanto, à medida que os caminhos se tornam mais longos, a diferença percentual de latência entre os dois métodos se torna cada vez mais pronunciada.

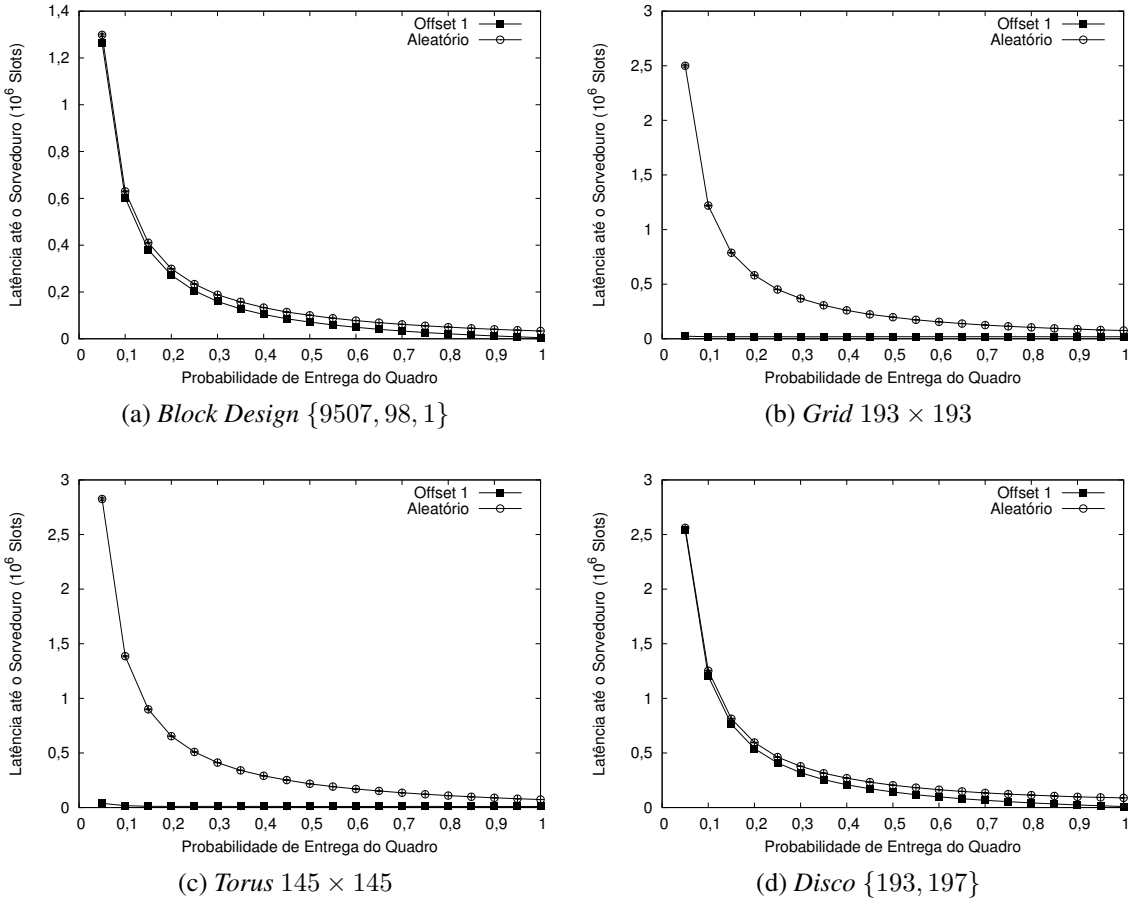
Já na Figura 7, são apresentados os resultados de latência fim-a-fim em função da probabilidade de entrega de quadros p, para uma comunicação em 7 saltos. É interessante notar que o *Grid* e o *Torus* apresentam uma latência quase linear para o *offset* = 1, uma vez que os dois métodos selecionam *slots* em linha (*i.e*, que se transformam em uma grande sequência de *slots* ativos no padrão linearizado). Quando a entrega de um quadro é feita com sucesso em um salto, o *slot* seguinte também será uma oportunidade de encontro no próximo salto. Embora isso seja válido para os quatro mecanismos estudados aqui (porque todos apresentam ao menos dois *slots* ativos consecutivos) no caso do *Grid* e do *Torus* com *offset* = 1 haverá até $n - 1$ oportunidades de encontro consecutivas. Ainda que a primeira tentativa não seja bem sucedida, os nós podem tentar novamente várias vezes em pouco tempo, mantendo a latência baixa mesmo quando p diminui. Este resultado é particularmente relevante porque a literatura acerca destes métodos consistentemente mostra que o *Block Design* é superior aos demais métodos em termos de NDT. No entanto, como ilustrado nestes resultados, no caso da latência fim-a-fim, *Grid* e *Torus* se mostraram a melhor opção para redes com enlaces de qualidade baixa, desde que o *offset* = 1 seja garantido entre os nós vizinhos. Já o *Block Design* e o *Disco* apresentam uma latência aumentando exponencialmente à medida que p cai, de forma similar ao que ocorre com o *offset* aleatório. Ainda assim, mesmo para estes mecanismos, o uso do *offset* = 1 resultou em redução na latência fim-a-fim para todos os valores de probabilidade avaliados.

5. Conclusão

Este artigo apresenta uma proposta para redução da latência de comunicação em múltiplos saltos dos mecanismos de *duty cycle* assíncrono baseados em *schedule*. Para tanto, consideramos a possibilidade de existência de um protocolo de sincronização dos nós em resolução de *slots* que permita que nós vizinhos sigam escalonamentos com *offsets* específicos. Este nível de sincronização pode ser obtido com baixo *overhead*, por exemplo, através da simples adição de um campo numérico nos pacotes de controle de um protocolo de roteamento informando o *slot* de tempo em que o nó atual se encontra. Com base nestas informações e no conhecimento do próximo salto em direção ao sorvedouro, nós podem alinhar seus padrões de *duty cycle*, possivelmente adicionando um *offset* entre eles.

Mostramos que o uso de um *offset* = 1 proporciona uma redução representativa da latência de comunicação em múltiplos saltos, ao mesmo tempo em que garante que novos nós podem ser facilmente adicionados à rede, sem a necessidade de um processo de sincronização prévia ou de uma fase inicial de sincronização na qual o novo nó é obrigado a permanecer com seu rádio ligado todo o tempo, resultando em desperdício de energia. Através de simulações numéricas, mostramos que, de fato, a introdução do *offset*

= 1 entre nós vizinhos resulta em reduções representativas de latência fim-a-fim. Estes ganhos se tornam ainda mais expressivos à medida que o número de saltos aumenta. Outro resultado interessante é o fato dos mecanismos *Grid* e *Torus* manterem latências quase lineares em relação à probabilidade de entrega de quadros dos enlaces na comunicação de múltiplos saltos com o uso do *offset* = 1, resultando em latências bem mais baixas que o *Block Design*, por exemplo (um mecanismo conhecidamente superior em termos de NDT [Carrano et al. 2014a]).



(a) *Block Design* $\{9507, 98, 1\}$ (b) *Grid* 193×193

(c) *Torus* 145×145 (d) *Disco* $\{193, 197\}$

Figura 7. Comparação da latência de comunicação para 7 saltos obtida com *offset* aleatório e igual a 1 por padrões baseados em *Block Design*, *Grid*, *Torus* e *Disco* variando-se a probabilidade de entrega do quadro (p).

Embora a avaliação tenha sido discutida sob a forma de simulações numéricas, futuramente pretende-se implementar este protocolo de sincronização de baixa resolução com escalonamento de *offsets* para corroborar os resultados obtidos e para permitir a avaliação do impacto do *overhead* de sincronização na comunicação de rede e no consumo energético dos nós. Também pretende-se aprofundar mais simulações e experimentos com outros valores fixos de *offset* entre nós vizinhos.

Referências

Anastasi, G., Conti, M., Di Francesco, M., and Passarella, A. (2009). Energy conservation in wireless sensor networks: A survey. *Ad hoc networks*, 7(3):537–568.

Bachir, A., Dohler, M., Watteyne, T., and Leung, K. K. (2010). MAC essentials for wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 12(2):222–248.

Carrano, R. C., Passos, D., Magalhães, L. C., and Albuquerque, C. V. (2013). Nested block designs: Flexible and efficient schedule-based asynchronous duty cycling. *Computer Networks*, 57(17):3316–3326.

Carrano, R. C., Passos, D., Magalhães, L. C., and Albuquerque, C. V. (2014a). A comprehensive analysis on the use of schedule-based asynchronous duty cycling in wireless sensor networks. *Ad Hoc Networks*, 16:142–164.

Carrano, R. C., Passos, D., Magalhães, L. C., and Albuquerque, C. V. (2014b). Survey and taxonomy of duty cycling mechanisms in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 16(1):181–194.

Choi, L., Lee, S. H., and Jun, J.-A. (2010). SPEED-MAC: Speedy and energy efficient data delivery MAC protocol for real-time sensor network applications. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–6. IEEE.

Dutta, P. and Culler, D. (2008). Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 71–84. ACM.

Jiang, J.-R., Tseng, Y.-C., Hsu, C.-S., and Lai, T.-H. (2005). Quorum-based asynchronous power-saving protocols for IEEE 802.11 ad hoc networks. *Mobile Networks and Applications*, 10(1-2):169–181.

Kandhalu, A., Lakshmanan, K., and Rajkumar, R. R. (2010). U-connect: a low-latency energy-efficient asynchronous neighbor discovery protocol. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 350–361. ACM.

Kruger, D., Pfisterer, D., and Fischer, S. (2010). CUPID-communication pattern informed duty cycling in sensor networks. In *2010 Fifth International Conference on Systems and Networks Communications*, pages 70–75. IEEE.

Link, J. Á. B., Wollgarten, C., Schupp, S., and Wehrle, K. (2011). Perfect difference sets for neighbor discovery: energy efficient and fair. In *Proceedings of the 3rd Extreme Conference on Communication: The Amazon Expedition*, page 5. ACM.

Lu, G., Krishnamachari, B., and Raghavendra, C. S. (2004). An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, page 224. IEEE.

Maekawa, M. (1985). An algorithm for mutual exclusion in decentralized systems. *ACM Transactions on Computer Systems (TOCS)*, 3(2):145–159.

Paruchuri, V., Basavaraju, S., Durresi, A., Kannan, R., and Iyengar, S. S. (2004). Random asynchronous wakeup protocol for sensor networks. In *Broadband Networks, 2004. BroadNets 2004. Proceedings. First International Conference on*, pages 710–717. IEEE.

Polastre, J., Hill, J., and Culler, D. (2004). Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107. ACM.

Rajendran, V., Obraczka, K., and Garcia-Luna-Aceves, J. J. (2006). Energy-efficient, collision-free medium access control for wireless sensor networks. *Wireless Networks*, 12(1):63–78.

Rowe, A., Mangharam, R., and Rajkumar, R. (2008). RT-link: A global time-synchronized link protocol for sensor networks. *Ad Hoc Networks*, 6(8):1201–1220.

Schurgers, C., Tsiatsis, V., Ganeriwal, S., and Srivastava, M. (2002). Optimizing sensor networks in the energy-latency-density design space. *IEEE transactions on mobile computing*, 1(1):70–80.

Sivrikaya, F. and Yener, B. (2004). Time synchronization in sensor networks: a survey. *IEEE network*, 18(4):45–50.

Sun, Y., Gurewitz, O., and Johnson, D. B. (2008). RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 1–14. ACM.

Tseng, Y.-C., Hsu, C.-S., and Hsieh, T.-Y. (2003). Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks. *Computer Networks*, 43(3):317–337.

Ye, W., Heidemann, J., and Estrin, D. (2002). An energy-efficient MAC protocol for wireless sensor networks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1567–1576. IEEE.

Zheng, R., Hou, J. C., and Sha, L. (2003). Asynchronous wakeup for ad hoc networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 35–45. ACM.

Trilha Principal do SBRC 2017
Sessão Técnica 17
Gerência de Redes e Sistemas P2P

Estudo sobre características de administradores de redes de computadores no Brasil para identificação e elaboração de personas

Hélio T. Oliveira[1], **Luciana Zaina**[1], **Leobino N. Sampaio**[2], **Fábio L. Verdi**[1]

[1] LERIS – Universidade Federal de São Carlos (UFSCar)
Sorocaba – SP – Brasil

[2]Dep. de Ciência da Computação – Universidade Federal da Bahia (UFBA)
Salvador – BA – Brasil

{helio.tibagi,lzaina,verdi}@ufscar.br, leobino@ufba.br

Abstract. *Although computer network monitoring has evolved rapidly in recent years, few advances have been made in the area of interface and interaction in network management systems. The set of new requirements, resulting from the expansion and diversification of users, requires studies that identify with more accurately the characteristics of their users. This article presents a study - collect and analysis - based on a field survey involving 70 Brazilian users with different profiles. As a result, the study identified two Personas, an artifact that describes characteristics and user needs, which can be used in the development of future interactive Internet monitoring tools.*

Resumo. *Apesar do monitoramento de redes de computadores ter evoluído rapidamente nos últimos anos, poucos avanços foram percebidos na área de interface e interação nos sistemas de gerência de redes. O conjunto de novos requisitos, resultantes da ampliação e diversificação dos usuários, exige estudos que identifiquem de forma mais precisa as características dos seus utilizadores. Este artigo apresenta um estudo - coleta e análise - realizado a partir de uma pesquisa de campo que envolveu 70 usuários brasileiros com diferentes perfis. Como resultado, o estudo identificou duas Personas, artefato que descreve caraterísticas e necessidades de usuários, que pode ser usado no desenvolvimento de futuras soluções interativas de ferramentas de monitoramento da Internet.*

1. Introdução

Desde o começo das suas operações, no início da década de setenta, a Internet sempre esteve acompanhada de atividades de monitoramento. No princípio, o grupo dos principais interessados em tais atividades era formado predominantemente por especialistas da área de redes, sobretudo, pesquisadores e engenheiros responsáveis pelo planejamento e operação da rede. Contudo, a expansão e consolidação da Internet comercial, o aumento da capacidade dos enlaces e o surgimento de diferentes tecnologias de transmissão trouxeram uma maior diversificação do grupo de interessados em atividades de monitoramento nos últimos dez anos [Rocha et al. 2016, Swany and Calyam 2014].

O monitoramento de redes passou a ser usado no apoio a diferentes atividades, tais como: suporte a engenharia de tráfego, planejamento de capaci-

dade, suporte a aplicações avançadas, verificação de SLA, segurança e redes experimentais [Swany and Calyam 2014]. Como consequência, diferentes aplicações, ferramentas e plataformas de medições tem surgido a cada dia [Rocha et al. 2016, Bajpai and Schönwälder 2015]. A maioria de tais iniciativas consiste em ambientes de operação e gerenciamento de rede, desenvolvidos para múltiplos propósitos, sem considerar o atual diversificado conjunto de perfis de usuários.

Apesar da existência de diferentes iniciativas de monitoração da Internet no Brasil [Rocha et al. 2016], a comunidade de redes também carece de projetos que levem em conta as demandas e características dos seus usuários. Muitas das aplicações e ferramentas utilizadas poderiam ser melhor projetadas se os desenvolvedores tivessem consciência sobre quem são os usuários, sobretudo nas questões relacionadas às funcionalidades de interação com os sistemas. Embora os usuários de sistemas de gerenciamento tenham formação técnica e com isto não tenham dificuldades ao interagir com uma interface, um mau projeto de interação pode causar problemas na eficiência e eficácia de suas tarefas, bem como dificultar a tomada de decisões baseada na leitura de informações apresentadas pelo sistema [Rogers et al. 2015]. Conhecer e identificar as *Personas* desse domínio permite a visão clara das características e necessidades de um grupo de potenciais usuários. A técnica de elaboração de *Personas* tem por objetivo representar os potenciais usuários de uma aplicação, de modo que o desenvolvedor estabeleça uma conexão de entendimento e de empatia com os usuários finais reais [Rogers et al. 2015, Graham and Bachmann 2004]. Diante dessa percepção, relatos recentes da literatura apontam para a necessidade de maiores investigações sobre os problemas de interação apresentados pelas ferramentas de monitoramento centradas no humano [Guimarães et al. 2016, Rosado et al. 2016]. Contudo, poucos trabalhos têm sido propostos nesta direção, sobretudo do que diz respeito à identificação e exploração de *Personas* que representam os usuários alvos desses sistemas.

Diante de tais motivações, este artigo apresenta uma pesquisa de campo feita com usuários de sistemas de monitoramento de redes de computadores no Brasil. A investigação buscou identificar as características de tais usuários e suas impressões sobre os sistemas utilizados no gerenciamento e operação de redes. Para obter a maior amostra de dados possível fornecida por Administradores de Redes (ADR), uma abordagem simples e eficaz foi tomada para coletar as características do uso de Ferramentas de Monitoramento de Rede (FMR), criando um formulário online e direcionando para esse público alvo obtendo, ao total, 70 respostas durante os meses de junho e julho de 2016. Para uma maior formalização do perfil dos usuários encontrados, como resultado desse levantamento, duas *Personas* foram identificadas e elaboradas. As *Personas* descritas apresentam-se como uma contribuição importante, pois podem auxiliar o projeto e desenvolvimento de futuros sistemas de monitoramento.

Este artigo está organizado da seguinte forma. A Seção 2 apresenta um resumo dos trabalhos relacionados ao levantamento de perfis de usuários de plataformas e aplicações de gerenciamento e operação de redes, sobretudo no que diz respeito ao monitoramento. A Seção 3 detalha a pesquisa de campo realizada. A Seção 4 apresenta os resultados obtidos na pesquisa de campo, enquanto que a Seção 5 descreve as Personas identificadas a partir desse levantamento. Por fim, a Seção 6 apresenta as conclusões do trabalho e os trabalhos futuros.

2. Trabalhos Relacionados

Com objetivos similares aos apresentados neste artigo, algumas iniciativas da comunidade de redes têm envidado esforços no sentido de fazer um levantamento das características dos usuários de ferramentas de gerência e monitoramento de redes. A rede européia GÉANT, através de um grupo de trabalho voltado para verificação e monitoramento de desempenho de redes (SIG-PMV Géant), conduziu uma pesquisa de campo que envolveu operadores, gerentes e usuários de aplicações de monitoramento no sentido de identificar o perfil dos interessados em informações de desempenho de redes [Chown 2016]. A pesquisa obteve um total de 25 respostas, predominantemente de administradores de rede da Europa com seis anos ou mais de experiência. Os resultados revelaram que 68% dos respondentes optam pelo uso de ferramentas *OpenSource* para realizar o monitoramento de redes. Além disso, foi constatado que, nas atividades de gerenciamento, o monitoramento do tráfego e o uso do protocolo SNMP são as atividades mais realizadas. Essas constatações são semelhantes às encontradas no estudo descrito neste artigo, conforme poderá ser verificado na próxima seção; o que permite inferir que as contribuições do estudo realizado no Brasil podem também atender as necessidades de outros países. Questões relacionadas ao desempenho de redes também foram abordadas pelo SIG-PMV. Revelou-se o grande uso de softwares como o Wireshark e IPERF e descobriu-se também que a taxa de transferência e latência da rede são os principais alvo de medições de desempenho, onde o nível IP é principal alvo de verificações.

A Internet das Coisas (IoT – *Internet of Things*) também tem motivado estudos relacionados ao levantamento de perfis de usuários. Em [Thoma et al. 2014] foi feito um estudo com 61 respondentes da área acadêmica e do mercado, onde é avaliado qual é a semântica ou sentido dado pelas pessoas que utilizam IoT, buscando conhecer as camadas de aplicação e de transporte, dentre outras características. Algumas iniciativas observam novas necessidades com o advento das redes SDN (*Software Defined Networking*), onde a atenção é diferente do tradicional. Isolani [Isolani et al. 2015] propõem um ciclo de gerenciamento onde métricas específicas de redes SDN possam ser monitoradas, processadas e mostradas com visualização interativas, para que o administrador de redes então possa tomar decisões e fazer os ajustes necessários. Os trabalhos citados mostram uma preocupação em se observar a interação do usuário com os sistemas. Contudo, observa-se que nenhum deles aponta claramente as características e necessidades do público alvo, que é uma contribuição importante da proposta deste artigo.

Outra iniciativa semelhante foi conduzida por [Goodall 2009]. Os autores conduziram um experimento com 8 pessoas na área de *Visualization for Cyber Security* - VizSec comparando uma aplicação que exibe dados de forma visual com gráficos, muito comum em softwares de monitoramento. O estudo também utilizou outra ferramenta mais tradicional que mostra os dados de forma descrita em tabela. Como resultado, utilizando a visualização gráfica os usuários conseguiram cumprir as tarefas em menos tempo e foram mais assertivos. A importância da visualização gráfica é vista nos resultados deste trabalho, indo de encontro com o resultado desse experimento.

Uma vez que pesquisas similares à conduzida neste trabalho normalmente são fechadas e realizadas por *vendors* de ferramentas de monitoramento comerciais, até onde sabemos, este trabalho apresenta uma contribuição de carácter inédito.

3. Pesquisa de Campo: Entendendo o usuário final

Entender o usuário final, ou seja, aquele que utiliza a FMR para desempenhar suas tarefas do dia a dia se mostra necessário para uma abordagem centrada no usuário [Guo et al. 2011, Billestrup et al. 2014] cobrindo assim o seu perfil, comportamentos e necessidades. Portanto, uma pesquisa de campo foi realizada para a obtenção dessas informações e apresentá-las na Seção 4 deste trabalho.

Levantamos algumas informações com especialistas e pesquisadores da área de redes de computadores para validar essa necessidade de obtenção de dados, e nos ajudar a elencar os principais tópicos que distinguem o uso de FMR, pela variada gama de empresas de ADRs que atuam no mercado. Então, um formulário com 18 questões objetivas e 4 dissertativas foi elaborado em 5 categorias: (i) Informações Demográficas, (ii) ferramenta de monitoramento e suas funcionalidades, (iii) importância visual, (iv) notificações SMS e (v) outras informações. Em especial, para perguntar quais funções da FMR o respondente utiliza, foi deixado um campo aberto para tornar possível citá-las de forma livre. Isso fez-se necessário devido ao enorme número de funcionalidades existentes nas FMRs e a dificuldade de enumerá-las com precisão: de maneira genérica poderia esconder detalhes importantes, e de maneira muito específica poderia causar ambiguidades devido à mesma função possuir denominações diferentes em cada FMR. Assim, para avaliar o número de funções existentes, uma análise manual foi realizada, com a interpretação humana em cada resposta, agrupando as funcionalidades conforme repetições começam a ser encontradas.

Recrutamos os participantes alvo, administradores de redes atuantes no mercado, a responderem um formulário online divulgando-o em grupos de discussão na área de redes de computadores, como o GTER, Resd-l e também divulgada para outros profissionais conhecidos no nosso grupo de pesquisa, cuidadosamente selecionados. Nenhuma espécie de incentivo ou premiação foi oferecida ao respondente em troca de sua participação.

O formulário ficou disponível durante os meses de junho e julho de 2016 e obtivemos ao todo 70 respostas. Os participantes concordaram com um Termo de Consentimento Livre e Esclarecido (TCLE) que declarava que os dados da pesquisa não seriam divulgados nominalmente, apenas de maneira quantitativa e estatística, tornando então restrita a identidade do respondente apenas aos autores deste trabalho.

Para possibilitar a análise dos dados de uma maneira mais prática, as respostas foram importadas em um banco de dados relacional. Em seguida, as respostas foram lidas uma a uma em busca de problemas ou inconsistências com a realidade. Nessa etapa, um campo aberto de texto livre se demonstrou oportuno, pois o respondente pode deixar explicações e informações adicionais que auxiliaram no entendimento de suas respostas. Respostas inválidas foram desqualificadas, e no momento de fazer alguma análise, no seu escopo, o número de respondentes pode ter sido menor do que 70 pois somente as respostas válidas foram consideradas. E, finalmente, várias consultas SQL foram realizadas com esses dados, testando e experimentando cruzamentos e descobrindo correlações ente eles, elaborando então gráficos para uma melhor interpretação dos dados.

Após realizarmos estas tabulações, tornou-se possível a elaboração das Personas, que será descrita com mais detalhes na Seção 5 deste trabalho. A Figura 1 mostra as etapas da metodologia da pesquisa de campo.

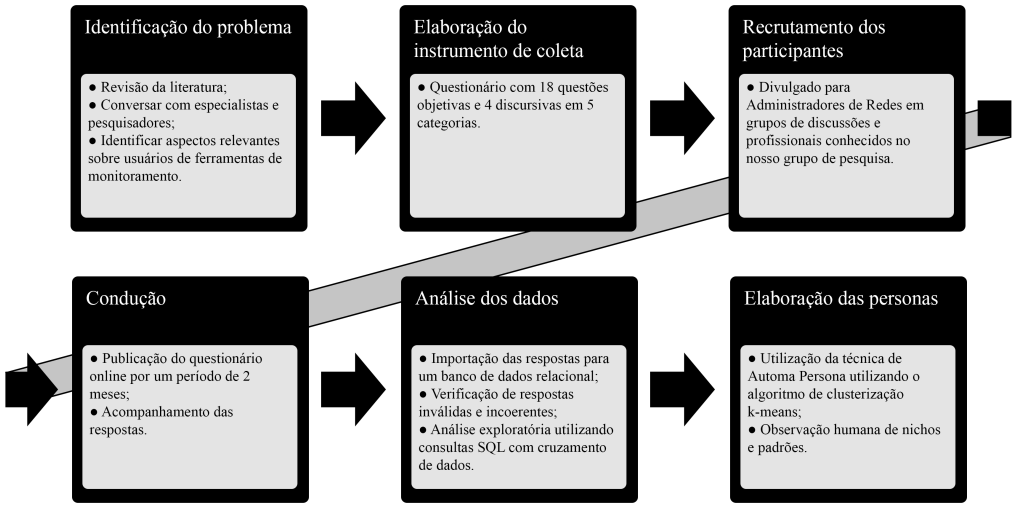


Figura 1. Metodologia da pesquisa de campo.

4. Análise dos Resultados

Esta seção apresenta análises considerando os dados coletados e organizados nos seguintes temas: informações demográficas dos participantes, características e funcionalidades sobre as FMR utilizadas pelos respondentes e sua importância em cada área da FMR e demais informações relevantes.

4.1. Informações Demográficas

O gráfico na Figura 2(a) mostra a experiência dos ADRs que responderam a pesquisa. Na Figura 2(b) o tamanho da empresa onde os respondentes atuam é apresentado. Para realizar a segmentação do tamanho da empresa, utilizou-se o algoritmo k-means [Hartigan and Wong 1979] implementado em uma *Stored Procedure* no banco de dados SQLServer. O k-means permite observar os dados agrupados em um determinado número de *clusters* desejados utilizando-se da distância euclidiana. Para a análise deste trabalho utilizou-se 3 *clusters* observando a quantidade de equipamentos de rede que a empresa possui, categorizando os agrupamentos pelo tamanho das empresas em: (i) pequena (de 0 a 200 equipamentos de rede), (ii) média (de 201 a 1000) e (iii) grande (mais de 1000 equipamentos de rede).

Em relação à experiência, observamos que o público respondente foi bastante heterogêneo, com uma distribuição bem uniforme nas faixas de anos de experiência. Ao analisar os gráficos, observamos que mais de 62% dos respondentes possui mais de 6 anos de experiência na área, sendo que 31,4% possui 10 anos ou mais, evidenciando que o grupo de ADRs que respondeu à pesquisa pode ser considerado um grupo experiente. Além disso, 74% dos respondentes pertence a empresas pequenas com no máximo 200 equipamentos de rede. Este último dado nos remete a deduzir que no Brasil não temos tantas empresas consideradas médias e grandes no que diz respeito ao número de elementos de rede. Apenas 3% dos que responderam à pesquisa estão em empresas com mais de 1000 equipamentos.

Os ADRs foram questionados a respeito de certificações relacionadas à área de monitoramento ou gerência de redes. Constatamos que a grande maioria dos adminis-

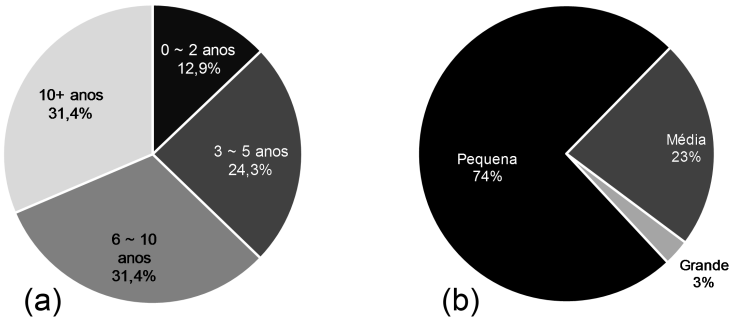


Figura 2. (a) Faixas de experiência de atuação na área de redes dos respondentes; (b) tamanho da empresa que atua.

tradores de rede que atuam no mercado, com 81,4%, não possui qualquer certificação na área de gerência ou monitoramento de redes. Dos que possuem, na sequência temos as certificações: 11,5% Zabbix, 2,9% CISCO, 1,4% Linux LPI, 1,4% Microsoft MCP e 1,4% Oracle.

4.2. Ferramenta de Monitoramento e suas Funcionalidades

O gráfico da Figura 3 apresenta a quantidade de citações de uso de uma determinada FMR. Constatou-se que o Zabbix é a FMR mais utilizada pelos ADRs, seguido pelo Cacti e o Nagios.

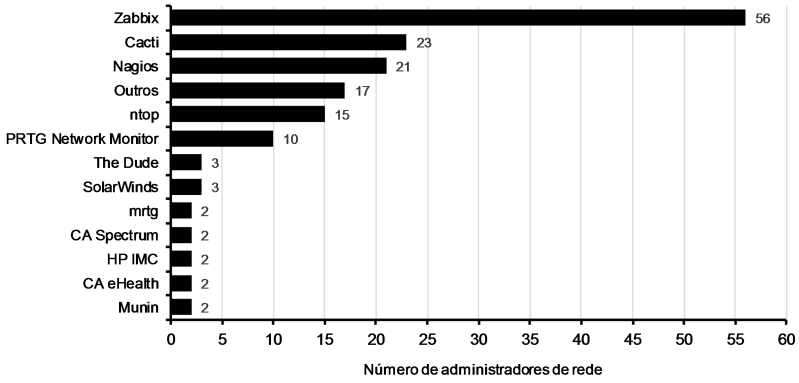


Figura 3. Softwares de monitoramento utilizados.

O gráfico na Figura 4 apresenta as funcionalidades de monitoramento e gerência de redes mais utilizadas pelos ADRs respondentes. As barras do gráfico estão segmentadas pela faixa de experiência a fim de observarmos a distribuição em relação à experiência dos ADRs. A Checagem via ICMP (ping), a obtenção de dados via SNMP e as medições de tráfico são as funcionalidades mais utilizadas pelos ADRs respondentes.

É possível constatar que ADRs com até dois anos de experiência predominantemente utilizam funções básicas da FMR, muitas vezes essas que são de fácil configuração. A checagem via ICMP (ping), e as medições de tráfego se destacam respectivamente com 4 e 5 citações. A conclusão é que um monitoramento mais simples é realizado, com 12 funcionalidades distintas.

Usuários na faixa de 3 a 5 anos de experiência passam a utilizar mais recursos, e alguns destes recursos possuem maior nível de complexidade de interpretação ou configuração. Ao invés de checagem por ICMP (bastante utilizada na faixa de até 2 anos de experiência), agora destaca-se o uso do protocolo SNMP que entrega mais informações detalhadas sobre o host ou equipamento de rede. É importante citar que medições de tráfego que estava em destaque na faixa de até 2 anos de experiência também pode ser realizada através dos dados vindos via SNMP. O uso de alertas para tomar ações de acordo com determinados gatilhos definidos pelo ADR também se destaca. A conclusão é que um monitoramento mais completo é realizado, com 15 funcionalidades distintas.

A amostra de usuários de 6 a 10 anos nos apresenta o uso de 20 funcionalidades, reflexo de um olhar mais maduro e diversificado no monitoramento. A conclusão é que alguns ADRs, além de praticar um monitoramento mais completo, passam a utilizar recursos mais específicos.

Por fim, ADRs com mais de 10 anos de experiência continuam utilizando as mesmas funcionalidades de monitoramento daqueles usuários menos experientes com uma ligeira diferença em algumas proporções. A checagem via ICMP (ping) volta a ser o destaque, demostrando que apesar de simples e muito usada por aqueles que não tem experiência, é também largamente utilizada por usuários experientes da área.

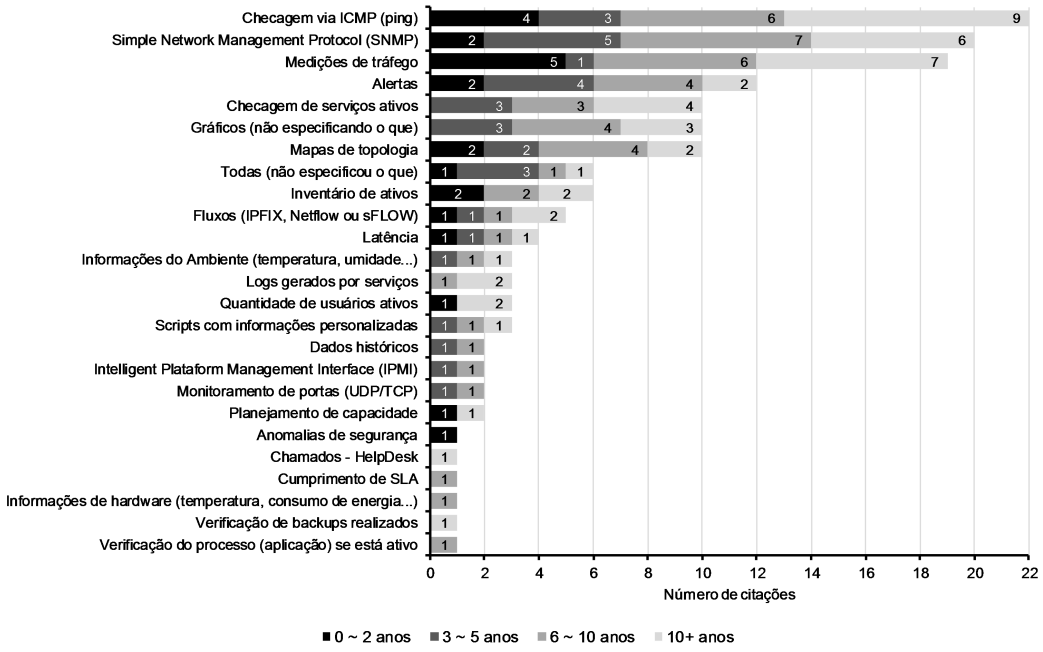


Figura 4. Funcionalidades utilizadas do software de monitoramento segmentadas por faixa de experiência.

Os ADR foram questionados sobre usar a FMR para fazer ajustes de desempenho em serviços que já estão funcionando de maneira satisfatória. As opções fornecidas foram: (i) “Sim”, (ii) “Não”, (iii) “Talvez, se estiver ao meu alcance” e (iv) “Não, não é a minha função”. A maioria dos administradores de rede utiliza a FMR não somente de maneira reativa, ou seja, em resposta a um determinado problema, mas também para melhorar serviços que já estão funcionando de maneira satisfatória. Os resultados mostraram que 60% responderam “Sim”. A segunda maior parcela com 32,9% escolheu “Talvez, se

estiver ao meu alcance". A opção "Não" ficou com 5,7% e, finalmente, a opção "Não, não é a minha função" ficou com 1,4%.

Os gráficos da Figura 5 mostram a quantidade de FMR utilizadas pelos ADRs. Pode-se concluir observando a Figura 5(b) que com a maior experiência do ADR, ele passa a utilizar mais FMRs. Também vale ressaltar que existe um *outlier* ao observamos quem utiliza 6 softwares de monitoramento, pois apresenta um valor atípico fugindo dos demais da série. Muitos ADRs usam somente uma ou duas ferramentas, como mostra a Figura 5(a), e a quantidade de ADRs que usam muitos softwares, tende a zero. Alguns usuários relataram na área livre que usam determinados softwares para determinadas funções, e outros dizem que preferem não confiar somente em uma FMR tendo assim uma espécie de redundância no monitoramento.

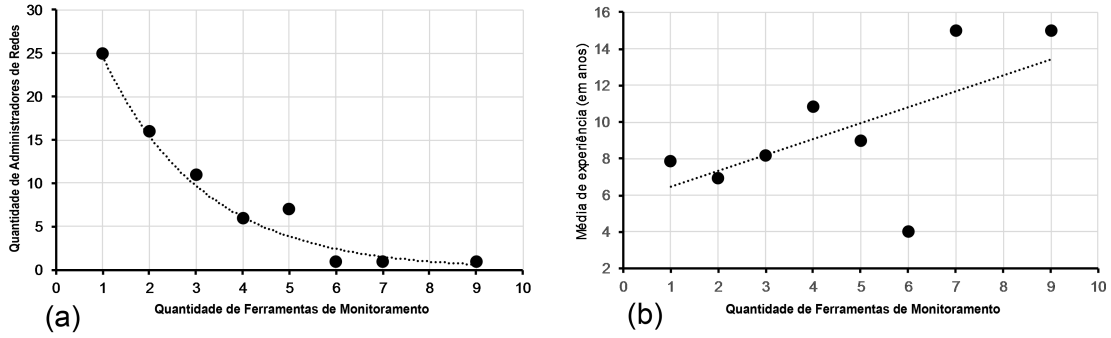


Figura 5. (a) Quantidade de ADR vs. quantidade de FMR utilizadas; (b) anos de experiência do ADR vs. quantidade de FMR utilizadas.

Os gráficos das Figuras 6 e 7 mostram a relação entre quantidade de equipamentos de rede e as funcionalidades utilizadas da FMR assim como a representação que cada funcionalidade teve na pesquisa dentro do seu grupo de tamanho de empresa.

Na Figura 6, podemos observar a maior variação de quantidade de equipamentos de rede nas funcionalidades de Ping, Medições de Tráfego e Obtenção de Dados via SNMP e que, juntamente a isso, essas funcionalidades são as mais utilizadas em pequenas empresas (percentual de representação na pesquisa). A funcionalidade de Mapa da Topologia possui uma faixa relativamente pequena, com no mínimo de 10 e máximo de 70 equipamentos, característica já prevista por serem empresas pequenas. A visualização de fluxos e scripts com informações personalizadas não são utilizadas por ADRs que cuidam de menos de 35 equipamentos de rede. Assim, podemos concluir que essa funcionalidade, mesmo em pequenas empresas não se demonstra aplicável se a administração é de poucos equipamentos, salvo aplicações específicas.

Na Figura 7, observamos a maior variação da quantidade de equipamentos de rede nas funcionalidades de obtenção de dados via SNMP e Ping, estas que também são as mais utilizadas em empresas médias. Em terceiro lugar na representatividade, com 31%, temos a medição de tráfego sendo utilizada em empresas com no mínimo 274 equipamentos e máximo de 500. A representatividade das demais funcionalidades fica em 13%.

Os dados obtidos nos gráficos das Figuras 6 e 7 fornecem parâmetros para o projeto de uma visualização gráfica em uma FMR, pois a volumetria de equipamentos segmentada em cada funcionalidade pode nos ajudar a projetar melhor a visualização

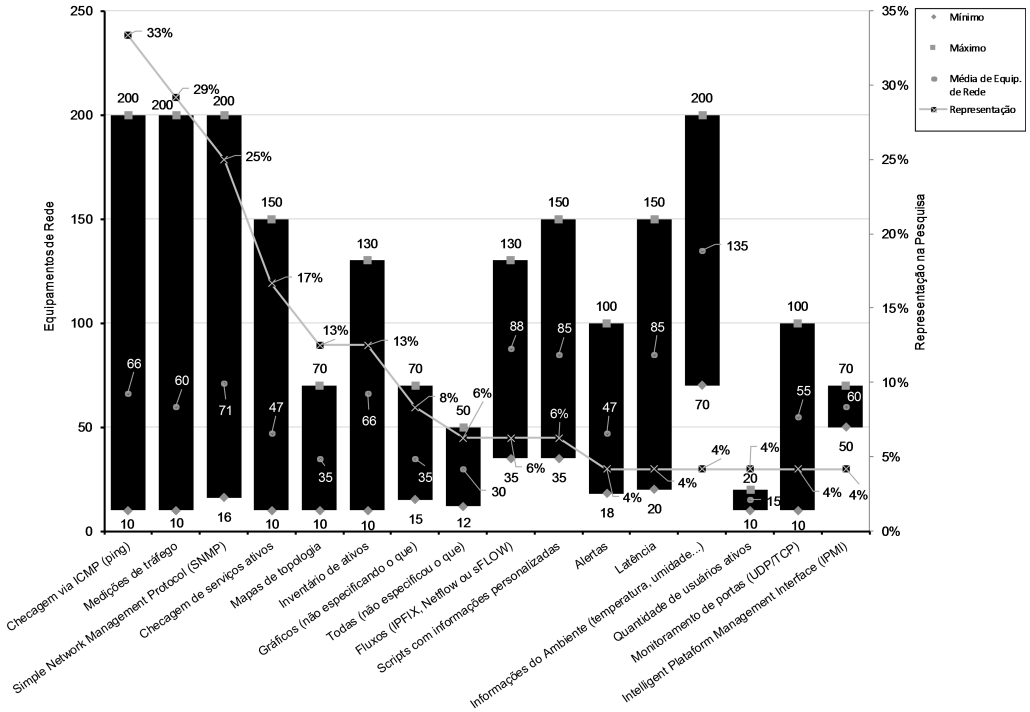


Figura 6. Equipamentos de rede vs. funcionalidades utilizadas no grupo de ”empresas pequenas”.

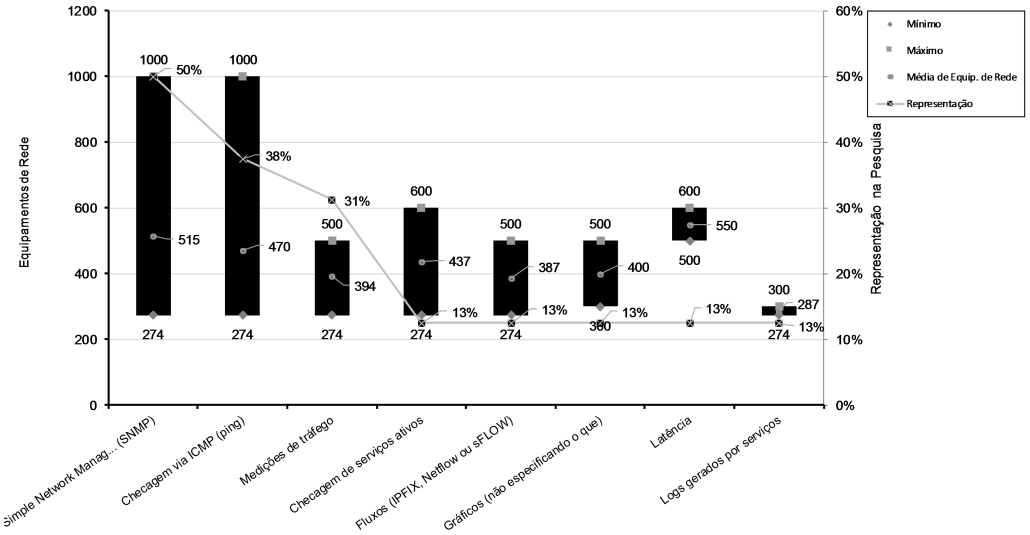


Figura 7. Equipamentos de rede vs. funcionalidades utilizadas no grupo de ”empresas médias”.

de uma FMR ou até mesmo adotar uma visualização escalável ideal [Shi et al. 2013, Taylor et al. 2000].

4.3. Importância Visual

No gráfico da Figura 8, podemos observar que, no geral, todas as características são definidas como importante ou muito importante, com destaque para a capacidade de mostrar a disponibilidade de equipamentos de rede sendo a mais citada como muito importante.

Na outra extremidade, a capacidade de definir permissões foi a mais citada como pouco importante.

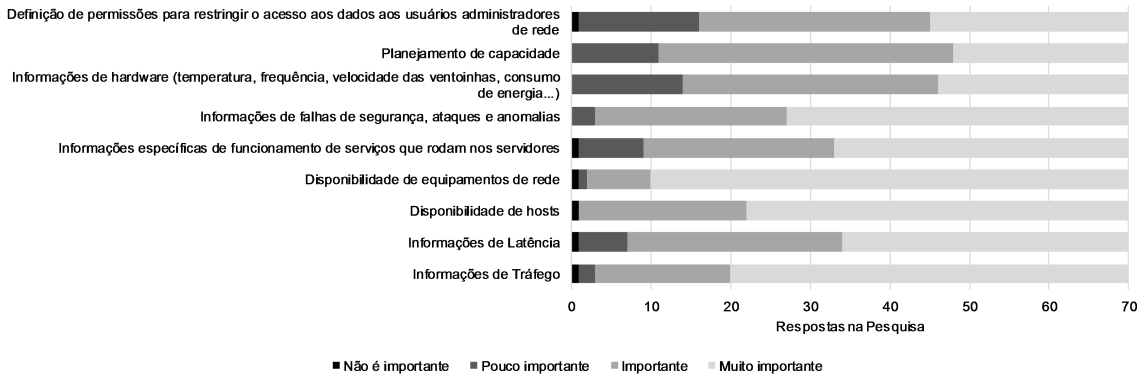


Figura 8. Grau de importância de características em um software de monitoramento quanto a visualização de dados.

4.4. Notificações SMS

O questionário também levantou dados sobre a possibilidade (interesse) de usar notificações por SMS ao detectar um problema. Neste sentido, metade dos respondentes afirmaram que usaria notificações por SMS enquanto que a outra metade afirmou que não usaria. Após um cruzamento simples entre os dados, constatamos que os ADRs que responderam que usariam SMS são responsáveis por gerenciar redes com maior número de equipamentos.

Os gráficos (a) e (b) na Figura 9 mostram em suas barras a quantidade de respostas que citam um determinado motivo pelo qual o ADR usaria ou não notificações por SMS.

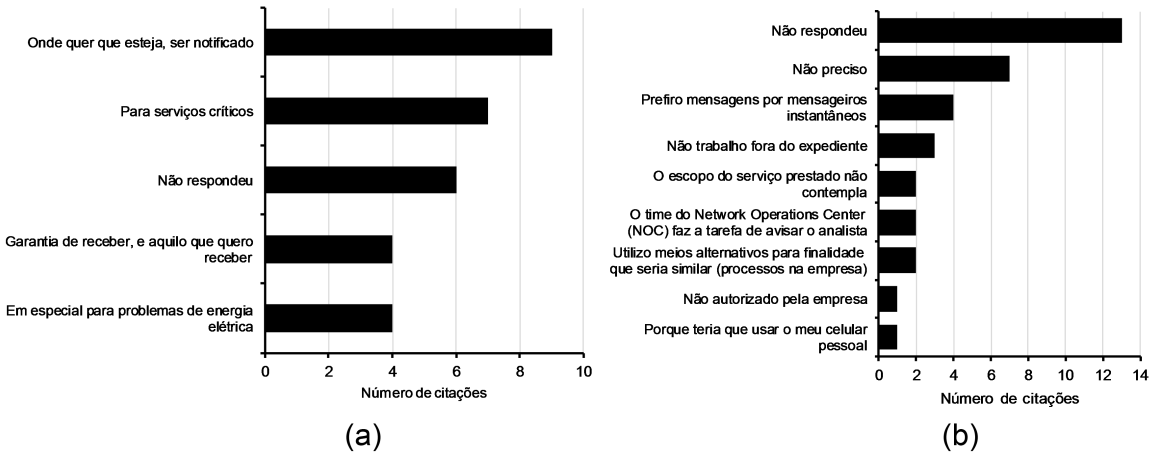


Figura 9. (a) Motivos pelo qual usaria notificações SMS; (b) motivos pelo qual não usaria notificações por SMS.

4.5. Outras Informações

Questionamos os ADRs a respeito da quantidade de usuários finais que utilizam as aplicações que funcionam na infraestrutura e cruzamos esse número com a segmentação

de tamanho de empresas: pequena, média ou grande. Os números mostraram que quanto maior a rede, em termos de quantidade de equipamentos de rede, maior o número de usuários finais. A média de usuários finais em empresas pequenas é de 838, em empresas médias é de 6.500 e, nas grandes, 15.0000.

Na Figura 10 (a) apresentamos o percentual dos ADRs que fazem a gestão de redes sem fio. O gráfico mostra que 72,9% dos respondentes realizam a gestão de redes sem fio, o que nos leva a concluir que é oportuno que a FMR possua funcionalidades específicas para gerenciamento de redes sem fio.

Na Figura 10 (b) apresentamos os dados a respeito do uso de redes SDN. A maioria dos usuários, com 52,9% não utiliza redes SDN. A segunda maior parcela com 38,6% não utiliza, entretanto gostaria de utilizar. Somando a parcela de usuários que já utilizam redes SDN com aquela que quer utilizar podemos visualizar que o advento das redes SDN é promissor e cobre quase a metade dos respondentes.

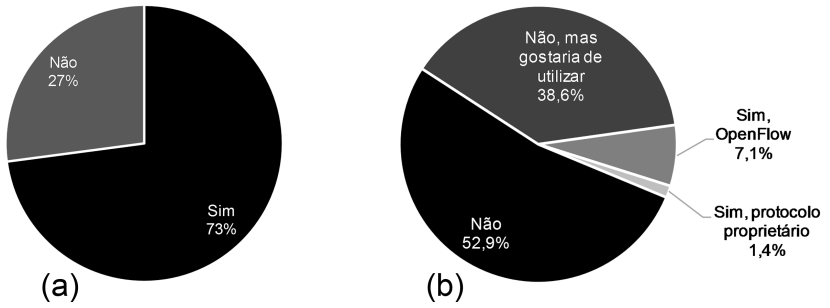


Figura 10. (a) Administração de redes sem fio; (b) utilização de redes SDN.

5. Identificando e Elaborando as Personas

Personas é uma técnica que permite descrever um grupo de usuários através de uma pessoa fictícia [Billestrup et al. 2014]. A descrição da persona é realizada por meio da análise dos dados coletados pela pesquisa realizada com grupos de usuários alvos. A técnica é muito utilizada por empresas em desenvolvimento de software para obter um melhor entendimento dos usuários que estarão utilizando o sistema. Além disso, a técnica quando utilizada durante o desenvolvimento de software, permite ao desenvolvedor visualizar com mais clareza os aspectos relevantes do grupo de usuários alvos da aplicação [Gothelf and Seiden 2013, Grudin and Pruitt 2002]. É importante destacar que uma persona não deve identificar características de um único usuário, pois seu principal objetivo é que o desenvolvedor tenha ciência das necessidade e características de um grupo alvo.

Existem diversas formas para representar uma persona. Para este trabalho utilizou-se do modelo proposto por Gothelf e Seiden [Gothelf and Seiden 2013], pois este apresenta um formato enxuto e direto para reportar as características relevantes. A partir do modelo, uma persona é representada por informações separadas em 4 quadrantes: (i) quadrante I - o nome fictício do usuário e sua foto. Dessa maneira é possível mencioná-lo e lembrar visualmente dele; (ii) quadrante II - informações demográficas que ilustrem o que é relevante no perfil do grupo de usuários; (iii) quadrante III - os principais comportamentos apresentados pelo grupo de usuários, como as atividades corriqueiras e recorrentes;

(iv) quadrante IV - as principais necessidades e objetivos do grupo, ou seja, um usuário deste grupo busca para ou como pretende cumprir as suas tarefas.

Baseando-se na análise dos dados e no modelo de persona descrito, realizou-se um mapeamento por quadrante das questões que poderiam contribuir para a elaboração das personas. A Tabela 1 apresenta esse mapeamento. Aplicou-se os dados da pesquisa para a elaboração das personas através do processo de Automa-Persona [Masiero et al. 2013] que consiste na utilização de um algoritmo de clusterização para separar as características da Persona.

Tabela 1. Mapeamento das questões da pesquisa

Quadrante	Questões Utilizadas
I	Nome (masculino ou feminino) do respondente.
II	Tempo de experiência; empresa onde atua; certificações; quantidade de equipamentos de redes e; quantidade de usuários finais na empresa.
III	Ferramentas de monitoramento utilizadas; como utiliza a ferramenta de monitoramento; opinião sobre notificações SMS; funcionalidades usadas da ferramenta de monitoramento.
IV	Importância visual de cada característica da ferramenta de monitoramento; funcionalidades utilizadas; texto livre de considerações finais.

A ideia é que cada Persona seja focada em um conjunto único de tarefas e informações, através da segmentação dos dados de volumetria, agrupando repetições de características utilizando consultas SQL e cruzando com o tempo de experiência e tamanho da empresa. Com isso, foi possível observar uma distinção de um ADR mais experiente em relação a um ADR iniciante no mercado, concluindo então que duas personas deveriam existir. A elaboração desse artefato chamado Persona muitas vezes pode resultar em algo diferente de uma tradicional segmentação de mercado [Guo et al. 2011], pois as Personas identificam comportamentos e atitudes dos usuários do sistema, para assim auxiliar o desenvolvedor a projetá-lo.

Como resultado do mapeamento descrito, foi possível identificar e elaborar as personas da Figura 11: Rômulo da Silva e Vinícius Moreira. As personas elaboradas representam perfis diferentes do público alvo: Rômulo representa um profissional mais experiente, enquanto que Vinícius representa características vistas em administradores de redes menos experientes.

6. Conclusão e Trabalhos Futuros

O monitoramento de redes é uma tarefa essencial para administradores de redes e engloba o uso de diversificada gama de opções de ferramentas, aplicações e perfis de usuários. Trata-se de uma atividade que tornou-se mais completa e complexa quando voltada para resolução de problemas avançados, gerenciamento de redes e, até mesmo, garantia de segurança. Os resultados obtidos a partir da análise da pesquisa de campo evidenciaram

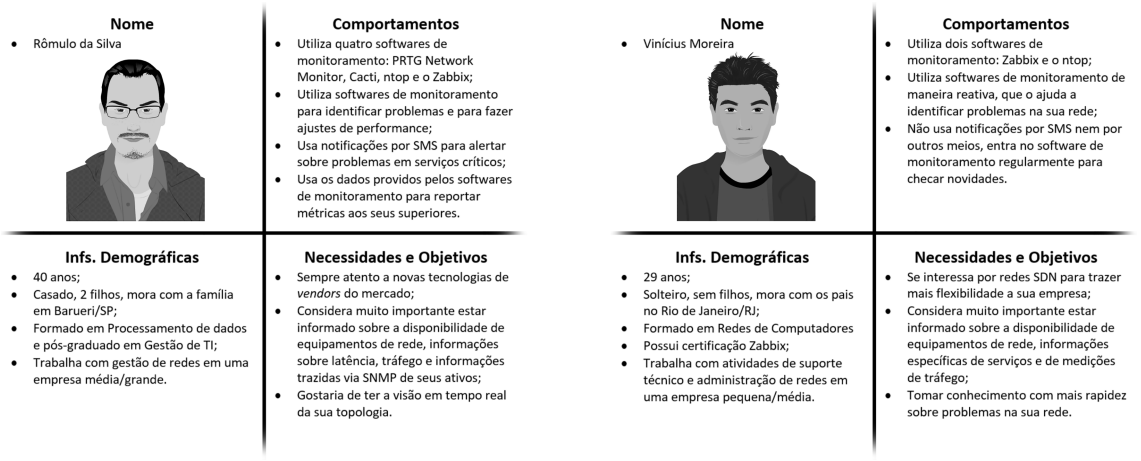


Figura 11. As duas Pesonas propostas de acordo com os resultados da pesquisa.

diferentes usos de protocolos de monitoramento, tais como o SNMP e o ICMP, para realizar o monitoramento de redes. Foram analisadas diversas ferramentas de monitoramento da atualidade, comportamentos e necessidades distintas, conforme o tamanho de necessidades das organizações sob avaliação. A partir deste estudo, foi possível elaborar duas Personas, que representam grupos de usuários menos e mais experientes, destacando os seus perfis, necessidades e comportamentos que contribuem para o projeto de interfaces e funcionalidades de interação.

Entre os principais desdobramentos deste trabalho, está a realização de um estudo focado nos ADRs a partir da realização de um experimento em laboratório. Neste estudo, serão verificados os erros mais comuns na utilização de ferramentas de monitoramento de redes e os problemas de interface relacionados à visualização de dados, dentre outros aspectos. Estes experimentos com os ADRs já estão sendo realizados e, após isso, pretende-se analisar as deficiências encontradas na FMR e então atuar na melhoria desta através do uso de técnicas de visualização de dados.

Agradecimentos
Os autores agradecem o suporte financeiro concedido pela FAPESP e CNPq, assim como agradecem também a todos os respondentes que participaram da pesquisa de campo realizada.

Referências

Bajpai, V. and Schönwälder, J. (2015). A survey on internet performance measurement platforms and related standardization efforts. *IEEE Communications Surveys Tutorials*, 17(3):1313–1341.

Billestrup, J., Stage, J., Nielsen, L., and Hansen, K. (2014). *Persona Usage in Software Development: Advantages and Obstacles*. IARIA XPS Press.

Chown, T. (2016). SIG-PMV Survey Overview. `https://wiki.geant.org/download/attachments/59933695/sig-pmv-survey-overview.pdf`. [Acessado em novembro de 2016].

Goodall, J. R. (2009). Visualization is better! a comparative evaluation. In *2009 6th International Workshop on Visualization for Cyber Security*, pages 57–68.

Gothelf, J. and Seiden, J. (2013). *Lean UX: Applying Lean Principles to Improve User Experience*. Lean series. O'Reilly Media, Incorporated.

Graham, D. and Bachmann, T. (2004). *Ideation: The Birth and Death of Ideas*. Wiley.

Grudin, J. and Pruitt, J. (2002). Personas, participatory design and product development: An infrastructure for engagement. In *PDC*, pages 144–152.

Guimarães, V. T., Freitas, C. M. D. S., Sadre, R., Tarouco, L. M. R., and Granville, L. Z. (2016). A survey on information visualization for network and service management. *IEEE Communications Surveys Tutorials*, 18(1):285–323.

Guo, F. Y., Shamdasani, S., and Randall, B. (2011). *Creating Effective Personas for Product Design: Insights from a Case Study*, pages 37–46. Springer Berlin Heidelberg, Berlin, Heidelberg.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.

Isolani, P. H., Wickboldt, J. A., Both, C. B., Rochol, J., and Granville, L. Z. (2015). Interactive monitoring, visualization, and configuration of openflow-based sdn. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 207–215.

Masiero, A. A., de Carvalho Destro, R., Curioni, O. A., and Aquino Junior, P. T. (2013). *Automa-Persona: A Process to Extract Knowledge Automatic for Improving Personas*, pages 61–64. Springer Berlin Heidelberg, Berlin, Heidelberg.

Rocha, A., Ziviani, A., Vieira, A. B., Sampaio, L., and Wehmuth, K. (2016). Revisitando metrologia de redes: Do passado às novas tendências. In *SBRC 2016 - Minicursos*, pages 151–209.

Rogers, Y., Sharp, H., and Preece, J. (2015). *Interaction Design: Beyond Human - Computer Interaction*. Wiley, 4th edition.

Rosado, W., Sampaio, L., and Monteiro, J. A. S. (2016). Recomendações de características ergonômicas para interfaces de sistemas de monitoramento de redes baseada em critérios de usabilidade. In *SBRC 2016*, pages 995–1008, Salvador, Bahia.

Shi, L., Liac, Q., Sun, X., Chen, Y., and Lin, C. (2013). Scalable network traffic visualization using compressed graphs. In *2013 IEEE International Conference on Big Data*, pages 606–612.

Swany, M. and Calyam, P. (2014). Second NSF workshop on perfSONAR-based multidomain network performance measurement and monitoring. Technical report, NSF.

Taylor, D. J., Halim, N., Hellerstein, J. L., and Ma, S. (2000). *Scalable Visualization of Event Data*, pages 47–58. Springer Berlin Heidelberg, Berlin, Heidelberg.

Thoma, M., Braun, T., Magerkurth, C., and Antonescu, A. F. (2014). Managing things and services with semantics: A survey. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–5.

Neutralidade de Rede com Modelos de Alocação de Banda e Comportamentos G-BAM – Análise de Compatibilidade

David S. S. Barreto[1], Rafael Freitas Reale[1,2], Joberto S. B. Martins[1]

[1] Universidade Salvador (UNIFACS) Salvador – BA – Brasil

[2] Instituto Federal da Bahia (IFBA) Campus Valença – BA – Brasil

davidssb@hotmail.com, reale@ifba.edu.br, joberto.martins@unifacs.br

Abstract. *Network neutrality (net neutrality) is an important management aspect of network infrastructures providing "services". Net neutrality is particularly relevant for Internet Service Providers - ISPs due to the ever increasing and critical use of "networks", internet included, as a commodity asset by our society. In brief, net neutrality aims isonomy in network treatment for Internet service delivery. This paper introduces the net neutrality and reasonable traffic management concepts and simulates the MAM, RDM and AllocTC-Sharing behaviors towards an analysis and evaluation of the effectiveness and compliance of using BAMs (Bandwidth Allocation Models) for net neutrality. From the technical point of view, using BAM as a traffic management practice aims to provide an "intelligent" isonomy for packet processing approach independently of its content. The results indicate that the AllocTC-Sharing behavior is appropriate and compatible with the imposed net neutrality rules.*

Resumo. *A neutralidade de rede é um importante aspecto de gerência das infraestruturas de rede que, de maneira geral, proveem "serviços". A neutralidade é particularmente relevante para as prestadoras do serviço de acesso à internet (ISPs). Isso na medida em que as "redes", a internet inclusa, são nos dias atuais um insumo de uso crítico e crescente por parte da sociedade. Em resumo, a neutralidade de rede visa um atendimento isonômico aos serviços transportados pela rede. Este artigo discute o conceito de neutralidade de rede e de gerenciamento de tráfego razoável e simula a operação dos comportamentos MAM, RDM e AllocTC-Sharing (Alloc) visando a análise da efetividade e conformidade do uso de BAMs (Bandwidth Allocation Models) para a neutralidade de rede. Do ponto de vista técnico, a utilização de um BAM como mecanismo de gerenciamento de tráfego visa implementar uma isonomia "inteligente" para o tratamento do tráfego independentemente do seu conteúdo. Os resultados indicam que o comportamento AllocTC-Sharing é apropriado e compatível com as regras de neutralidade de rede impostas.*

1. Introdução

O termo *net neutrality*, amplamente difundido na literatura em língua inglesa, tem duas traduções no Brasil: "neutralidade da rede" [RAMOS 2015], no qual a palavra "rede" é associada à Internet; e "neutralidade de rede" [LEI Nº 12.965 2014], no qual "rede" é associada às estruturas de rede dos ISPs (*Internet Service Providers*) que fazem parte do

conglomerado de redes que é a Internet. Este trabalho opta por utilizar o termo "neutralidade de rede".

O debate em torno da neutralidade de rede tomou forma nos Estados Unidos da América e seguiu em outros lugares como Chile, Brasil e Europa dentro de um contexto de preocupações sobre os modelos de negócios dos grandes ISPs, que lidam com o aumento exponencial do volume de tráfego, diferentes tipos de serviços trafegados com diferentes requisitos de Qualidade de Serviço (QoS - *Quality of Service*) e a crescente oferta de conteúdo por empresas provedoras de serviços, aplicações e conteúdo (CAP - *Content and Application Provider*).

Umas das questões centrais deste debate era a habilidade dos ISPs praticarem tratamento diferenciado para o tráfego da Internet. Neste sentido, a fim de limitar a capacidade dos ISPs de interferir no tráfego de aplicações, conteúdos e serviços em suas redes, normativos de diversos países impuseram limites para o uso de práticas de gerenciamento de tráfego/rede, como inspeção de pacotes, *traffic shapping* e mecanismos de gerenciamento de congestionamento. Como efeito prático desses normativos, qualquer prática que vise o gerenciamento de recursos de rede (largura de banda, *buffer*, etc.) precisa atender aos limites estabelecidos.

O conceito de neutralidade de rede é amplo e envolve perspectivas econômicas, ideológicas, sociais e técnicas dos quatro principais atores desta discussão: os ISPs, os CAPs, os usuários finais e as Autoridades Reguladoras Nacionais - ARNs. As interações entre os três primeiros atores são fortemente governadas por interesses econômicos e cabe às ARNs estipular regras com o objetivo de preservar e promover a inovação e o crescimento econômico que a Internet possibilita, bem como manter o ambiente descentralizado de interação social, cultural e político [SCHEWICK 2015].

Não existe uma conformidade de definição para o termo neutralidade de rede. O termo tem sido utilizado de forma ampla para descrever o livre acesso à Internet, o que, para autores como Tim Wu [WU 2003], não é muito preciso. Wu e outros autores [JORDAN 2007], [BEREC 2011] utilizam este termo para caracterizar o tratamento igualitário de aplicações, conteúdos e serviços que trafegam pela internet e a importância da transparência nas práticas de gerenciamento de tráfego. A definição do Órgão Regulador da União Europeia [BEREC 2011] se apresenta como uma das mais simples e direta: "a neutralidade de rede é um princípio a partir do qual todos os pacotes são tratados de forma igual através de uma infraestrutura IP".

Embora esta seja uma regra de não discriminação, a discriminação de serviço é uma questão importante nestes normativos [SCHEWICK 2015], pois é preciso garantir a funcionalidade das diferentes aplicações que se utilizam da Internet. Tais aplicações têm uma grande variedade de requisitos de QoS e satisfazê-los adequadamente (num cenário de recursos escassos) sem aplicar diferenciação de serviço é uma tarefa no mínimo desafiadora. No contexto da neutralidade de rede, o problema a ser endereçado do ponto de vista dos ISPs é o atendimento aos diferentes requisitos de QoS das aplicações (principalmente em ocasiões de congestionamento) de forma compatível com as regras vigentes. Por outro lado, do ponto de vista dos usuários das redes, CAPs inclusos, o resultado esperado é o fornecimento do serviço com qualidade, sem interferência na escolha dos usuários e sem distorções na competição.

Este artigo propõe a utilização do Modelo de Alocação de Banda Generalizado (*Generalized Bandwidth Allocation Model* – G-BAM), especificamente com o comportamento (*behavior*) AllocTC-Sharing [REALE, NETO e MARTINS 2011], como uma prática de gerenciamento de tráfego aplicável a ISPs em redes MPLS/DS-TE (*Multiprotocol Label Switching with Traffic Engineering*) com o propósito de contribuir para a funcionalidade das diversas aplicações da Internet, dentro das regras de neutralidade de rede. Este comportamento agrega uma característica oportunista para a alocação de largura de banda que permite empréstimos mútuos entre Classes de Tráfego (CTs) de baixa e alta prioridade e só interfere na topologia de LSPs (*Label Switched Paths*) existentes em situações de conflitos resultantes da ausência de largura de banda (congestionamento).

O artigo está organizado da seguinte forma: Os trabalhos relacionados estão presentes na seção 2. Na Seção 3 é apresentado um breve descritivo do G-BAM e do comportamento AllocTC-Sharing. Na Seção 4 são apresentados os conceitos de neutralidade de rede e de gerenciamento de tráfego razoável. Na seção 5 é apresentada uma avaliação dos resultados obtidos através de simulação e, por fim, conclusões e trabalhos futuros são apresentados na Seção 6.

2. Trabalhos Relacionados

Alguns trabalhos oferecem reflexões a partir de uma abordagem técnica, explicando como a neutralidade de rede e discriminação de tráfego (baseada ou não em QoS) podem alcançar os mesmos objetivos e exploram possíveis soluções que envolvem a gestão de recursos de rede.

Wu [WU 2003] não apresenta uma solução técnica, mas mostra um princípio implementável de neutralidade de rede e descreve como seria o comportamento das operadoras neste contexto, além de refletir sobre questões de neutralidade de rede utilizando discriminação de tráfego associada à largura de banda. O autor acredita que o gerenciamento de largura de banda está alinhado com a neutralidade de rede. Certas classes de aplicações nunca vão funcionar corretamente a menos que largura de banda suficiente e QoS sejam garantidas. Ele também acredita que se as operadoras apenas gerenciarem a banda disponível o resultado é um ambiente mais competitivo.

Schewick [SCHEWICK 2015] propõe oito possíveis regras de neutralidade de rede. Uma abordagem específica dada por van Schewick permite a discriminação entre classes de aplicações que não são iguais. Esta regra permitiria que os ISPs tratassem classes de aplicações de forma diferente, desde que elas tratassem tipos de tráfegos similares da mesma forma, por exemplo, classificando-os dentro de uma mesma CT.

Enquanto Jordan [JORDAN 2007] propõe uma política de neutralidade de rede com a finalidade de proibir comportamento anti-concorrencial sem restringir formas desejáveis de gestão da rede, [WÓJCIK 2011] apresenta uma solução técnica para o problema da neutralidade utilizando uma arquitetura QoS. Segundo o autor, a operação da rede permaneceria neutra, ainda que oferecesse diferenciação de serviço.

3. Modelos de Alocação de Banda e o G-BAM

Em resumo, um BAM arbitra a alocação de largura de banda (recurso) para os usuários de uma rede IP/MPLS/DS-TE [REALE, MARTINS, *et al.* 2015]. Por usuário, entende-

se qualquer aplicação, serviço ou usuário final capaz de solicitar uma LSP para o tráfego de seus dados.

Existem 2 modelos de alocação de banda considerados mais básicos: MAM (*Maximum Allocation Model*) e RDM (*Russian Dolls Model*) [REALE, BEZERRA e MARTINS 2014]. O AllocTC-Sharing foi proposto em [REALE, NETO e MARTINS 2011] como mecanismo capaz de melhorar os níveis de atendimento ao tráfego. Cada modelo de alocação de banda trabalha com um conjunto de Classes de Tráfego (CTs), tipicamente entre 3 e 7. Existe uma prioridade definida para cada CT e, para cada CT é configurada a banda máxima que pode ser utilizada para as aplicações desta classe de tráfego. Esta banda máxima que o modelo pode alocar é denominada de restrição de banda - BC (*Bandwidth Constraint*) [LE FAUCHEUR e LAI 2003].

O que os diferentes modelos existentes implementam é a forma como a largura de banda disponível pode ser compartilhada entre as diferentes CTs. O modelo MAM assume que não existe compartilhamento. O modelo RDM permite o compartilhamento da banda não utilizada pelas CTs de alta prioridade por CTs de baixa prioridade (HTL – *High-to-Low*). O AllocTC-Sharing, por sua vez, permite que CTs de maior prioridade possam também utilizar a banda disponível e não utilizada pelas CTs de menor prioridade, lançando mão da estratégia de compartilhamento LTH (*Low-to-High*), em complemento ao HTL.

Assim sendo, cada modelo implementa efetivamente um "comportamento" na forma como a largura de banda é alocada e o modelo G-BAM (*Generalized BAM*), generaliza todos os possíveis comportamentos existentes num único modelo através de uma configuração de parâmetros. Ou seja, o G-BAM permite que qualquer comportamento BAM seja utilizado na gestão da alocação de banda na rede através de uma configuração adequada de parâmetros [REALE, BEZERRA e MARTINS 2014].

A operação geral do AllocTC-Sharing, foco desta proposta, pode ser resumida da seguinte maneira: (1) uma nova requisição de LSPs resulta em estabelecimento de LSP se houver banda disponível no enlace, com ou sem necessidade de compartilhamento (LTH e/ou HTL); (2) no caso de não existir largura de banda disponível no enlace, o algoritmo tenta devolver banda emprestada previamente a CTs de maior prioridade (devolução de LSP estabelecida) ou devolver banda emprestada previamente a CTs de menor prioridade (preempção de LSP estabelecida) até o valor extrapolado do respectivo BC, nesta sequência; (3) caso não exista banda suficiente a ser devolvida que possibilite o estabelecimento da LSP requisitada, a requisição de LSP é bloqueada.

4. Discriminação de tráfego no escopo da neutralidade de rede

A regra de não discriminação não proíbe a utilização de práticas de gerenciamento de rede, incluindo medidas de gerenciamento de tráfego. Algumas nações publicaram normativos sobre neutralidade de rede, como o Chile, Brasil, União Europeia, Japão, Singapura e Estados Unidos da América [WEBB e HENDERSON 2012]. Três regras majoritariamente comuns podem ser destacadas:

- Proibição de bloqueio de conteúdo legal, aplicativos e serviços;
- Permissão para utilização de mecanismos de gestão de rede/tráfego, desde que de forma razoável, em circunstâncias definidas; e

- Necessidade de divulgação de informações críticas relativas às práticas de gestão de rede, pelas operadoras.

Como dito, muitos normativos preveem a utilização de práticas ou mecanismos de gerenciamento de rede/tráfego desde que sejam "razoáveis" [BEREC 2016], [FCC 2015] ou "justificáveis" [DECRETO Nº 8.771 2016]. Estes mecanismos são considerados desvios (ou exceções) à regra de não discriminação e não são considerados como infração à neutralidade de rede.

No contexto da neutralidade de rede, o termo "gerenciamento de rede" refere-se a práticas cujo objetivo é manter, proteger e assegurar o funcionamento eficiente de uma rede. Segundo [BEREC 2016], o objetivo de um gerenciamento de tráfego razoável é contribuir para a utilização eficiente dos recursos da rede e para a otimização da qualidade global da transmissão, respondendo objetivamente aos diferentes requisitos de QoS de determinadas categorias de tráfego. Para se qualificar como gerenciamento de rede/tráfego "razoável" ou "justificável", a prática deve atender a certos requisitos, como: (1) alcançar um propósito de gerenciamento legítimo, (2) ser estreitamente desenhada para alcançar este propósito, (3) ser transparente, (4) ser não discriminatória, (5) ser proporcional, (6) ser baseada em considerações técnicas e não comerciais, (7) não monitorar o conteúdo dos pacotes, e (8) ser excepcional.

Assim sendo, a conformidade com a neutralidade de rede é estabelecida no atendimento tanto à sua regra de não discriminação quanto aos requisitos para qualificação como gerenciamento de tráfego razoável, de acordo com os seguintes perfis de operação:

- Tratamento igualitário do tráfego, como regra; e
- Aplicação de diferenciação de serviço de forma razoável, como exceção.

4.1. Classificação do tráfego

O mapeamento do tráfego em Classes de tráfego (CTs) é o primeiro passo em direção a uma operação de rede alinhada à neutralidade de rede. Do ponto de vista técnico, o mapeamento de aplicações e serviços em CTs visa proporcionar experiências satisfatórias ao usuário sob o ponto de vista de QoS e de QoE (*Quality of Experience*). Representa uma visão puramente técnica de como as aplicações devem usar os recursos de rede. Como exemplo, certas classes de aplicações precisam de um mínimo de largura de banda garantida, enquanto outras classes não. Além disso, algumas classes de aplicações aproveitam a disponibilidade de largura de banda para melhorar a experiência do usuário, adaptando suas taxas de transmissão.

São dois os aspectos principais a considerar em um mapeamento de aplicações em CTs alinhado à neutralidade de rede: primeiro deve-se observar os requisitos mínimos de QoS para cada tipo de tráfego; e segundo, aplicações com requisitos de QoS semelhantes tem que ser agrupadas numa mesma CT, com o objetivo de se garantir uma operação não discriminatória [BEREC 2016]. A classificação do tráfego é comumente realizada pelo ISP por seus próprios métodos. Está fora do escopo deste trabalho avaliar quais mecanismos seriam mais adequados, embora seja vedado monitorar o conteúdo dos pacotes (*Deep Packet Inspection* – DPI) [LEI Nº 12.965 2014].

A Tabela 1 apresenta uma proposta de mapeamento de aplicações em Classes de Tráfego baseada na associação de classes de QoS previstas na Recomendação ITU-T

Y.1541 [ITU-T 2011] com as capacidades de transferência previstas na Recomendação ITU-T Y.1221 [ITU-T 2010]. Por definição, CT2 tem maior prioridade de atendimento do que CT1 e CT0 tem a menor prioridade entre as CTs. A definição da quantidade de CTs e qual tipo de tráfego cada uma delas comporta é essencial na avaliação do caráter não discriminatório da operação da prática de diferenciação de serviço. Desta forma, a classificação de tráfego em três classes de tráfego pode atender aos requisitos de qualificação nº 2, 4 e 6 listados anteriormente.

É claro que a alocação de aplicações e serviços similares em uma CT específica não garante largura de banda para todas as aplicações. Os recursos da rede são, especialmente durante congestionamento, escassos e têm de ser "disputados". Neste ponto, a operação do G-BAM assumirá e apoiará a intervenção no tráfego (aplicação de prioridades para o acesso à largura de banda) de forma alinhada à neutralidade de rede. Em simulação utilizamos a classificação de tráfego proposta, i. e., tratamos classes de aplicações de forma diferente, mas tratamos tráfego semelhante de forma semelhante.

Tabela 1 – Mapeamento de Aplicações em Classes de Tráfego (CTs)

Tipo de Aplicação (exemplos)	CT	Tipo de Tráfego
Aplicações web, navegação web, e-mail, transferência de arquivos, outras de caraterística semelhante.	CT0	Tráfego melhor esforço (*best effort*).
Tempo real, sensíveis à variação do atraso (*jitter*), áudio e vídeo (*streaming*, conferência), aplicações interativas, outras de caraterística semelhante.	CT1	Tempo real e/ou de uso intensivo de largura de banda.
Backup, imagens médicas, realidade virtual, outras de caraterística semelhante.	CT2	Crítico e/ou de emergência.

5. Implementação e avaliação da neutralidade de rede com o uso do comportamento AllocTC-Sharing

5.1. Primeiras análises

O propósito do uso de um BAM no contexto deste trabalho é contribuir para a funcionalidade das diversas aplicações da Internet, sob a forma da busca pelo atendimento adequado aos requisitos de qualidade dos serviços e aplicações. Uma consequência da escolha do AllocTC-Sharing é a otimização do desempenho da rede em relação aos BAMs mais básicos. Segundo a [FCC 2015], é considerada legítima a busca pela otimização da performance global da rede e manutenção da qualidade de experiência dos usuários, considerando a variedade de tráfego sendo transportado pela rede. O G-BAM apenas gerencia a largura de banda disponível nos enlaces da rede. Assim, este mecanismo atende concomitantemente aos requisitos nº 2 e 7 para qualificação da prática como gerenciamento de tráfego razoável. O atendimento ao requisito nº 1 é avaliado por intermédio das simulações, detalhadas mais à frente.

Em relação à transparência requisitada (requisito nº 3), isto é, publicidade das práticas de gerenciamento de tráfego, entende-se que é uma obrigação dos ISPs e que seu objeto transcende questões puramente técnicas, como configuração e parâmetros utilizados. Exemplos de medidas de transparência estão listados em [DECRETO Nº 8.771 2016).

Para atender ao requisito nº 4, a operação do G-BAM com o comportamento AllocTC-Sharing deve atender tanto à regra de não discriminação, quanto intervir (excepcionalmente) no tráfego baseado no seguinte conjunto de condutas: (i) situações similares, em termos de requisitos de QoS, devem receber tratamento similar; e (ii) situações diferentes, também em termos de requisitos de QoS, devem receber tratamento diferentes. Para [BEREC 2016], tais condutas também implicam em tratamento não discriminatório. Os resultados das simulações indicam que o AllocTC-Sharing segue estritamente o esquema de classificação de tráfego, baseado na sensibilidade aos requisitos QoS das aplicações (Tabela 1). Esta operação, ao se basear apenas em considerações técnicas, atende ao requisito nº 6.

As simulações apresentadas a seguir avaliam o atendimento aos requisitos nº 1, 4, 5 e 8 para qualificação da prática como gerenciamento de tráfego razoável. Em especial, os experimentos demonstram que a operação do G-BAM com comportamento AllocTC-Sharing implica em tratamento igualitário do tráfego, como regra e implica em tratamento diferenciado, de forma excepcional e proporcional.

5.2. Simulações

As simulações foram realizadas em dois cenários distintos: o primeiro cenário utiliza geração aleatória de tráfego. O segundo cenário utiliza um perfil de geração de tráfego pré-estabelecido (não aleatório). É utilizado o simulador especializado BAMSim - *Bandwidth Allocation Model SIMulator* [NETO e MARTINS 2008] com o modelo G-BAM configurado de forma a reproduzir o comportamento dos modelos MAM, RDM e AllocTC-Sharing e da configuração FRFS (descrita mais à frente). Cada cenário foi simulado com 5 sementes aleatórias, sem reutilização, e os resultados[1] apresentados representam a média dos valores obtidos nas cinco simulações. É utilizada a topologia de rede NSF (*National Science Foundation*), que contém 14 nós e 42 enlaces bidirecionais (Figura 1).

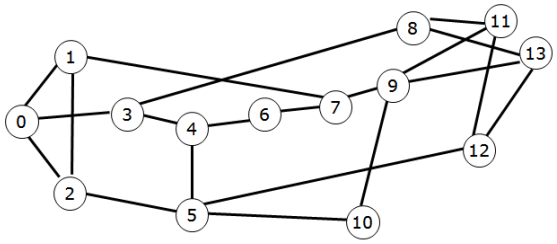


Figura 1 – Rede NSF

Os parâmetros de configuração comuns aos dois cenários são os seguintes:

- Capacidade dos enlaces: 1Gbps
- Classes de tráfego – CTs existentes: CT0, CT1 e CT2 (Tabela 1)
- Limites de empréstimo HTL e LTH: 100%
- Restrições de banda (*Bandwidth Constraints* - BCs): BC0, BC1 e BC2, de acordo com a Tabela 2.

[1] Para validação geramos os intervalos de confiança (95%) e desvios padrões. Resultados completos: http://www.rafaelreale.net/artigos/NeutralidadeSBRC/NeutralidadeSBRC.xlsx

Tabela 2 – Restrições de banda por Classe de Tráfego (CT)

BCs	TCs	BC (%)	MAX BC – MAM (Mbps)	MAX BC – RDM (Mbps)	MAX BC – Alloc (Mbps)
BC0	CT0	25	250	1000	1000
BC1	CT1	35	350	750	1000
BC2	CT2	40	400	400	1000

5.2.1. Primeiro Cenário de Simulação – Configuração e resultados

Nesta subseção são apresentados os resultados que comprovam que o G-BAM com comportamento AllocTC-Sharing contribui para a funcionalidade das diversas aplicações por duas razões: leva em conta seus requisitos de QoS e otimiza o desempenho da rede. Adicionalmente, os resultados indicam que a diferenciação de serviço aplicada pelo AllocTC-Sharing é a opção menos interferente capaz de alcançar estes dois resultados, sendo, portanto, uma solução proporcional.

A avaliação dos níveis de otimização do desempenho da rede é realizada por intermédio de comparação entre os comportamentos MAM, RDM e AllocTC-Sharing. São utilizadas as mesmas métricas de desempenho empregadas em [REALE, NETO e MARTINS 2011]: (1) largura de banda atendida, (2) quantidade de LSPs bloqueadas, (3) quantidade de LSPs preemptadas, e (4) quantidade de LSPs atendidas. Por outro lado, a avaliação do comportamento AllocTC-Sharing quanto a ser uma solução proporcional é realizada por intermédio de comparação com uma configuração sem Classes de Tráfego, conforme sugerido em [BEREC 2016].

Nesta simulação de 24 horas de duração, todos os nós da topologia utilizada são possíveis fontes de tráfego e possíveis destinos. Desta forma, qualquer um dos 42 enlaces pode experimentar momentos de subutilização do enlace, alta ocupação e/ou congestionamento prolongado, com variações aleatórias de tráfego entre as CTs. Os parâmetros configurados para cada LSP são os seguintes:

- Tempo de vida das LSPs – modelagem exponencial – média de 300 segundos;
- Carga das LSPs – Distribuição aleatória entre 5 Mbps e 35 Mbps;
- Critério de parada – 24 horas (86.400s); e
- Intervalo entre chegadas de solicitações de LSPs modelado exponencialmente tendo como média LSPs geradas a cada 2 segundos.

Resultados - Otimização do desempenho da rede

A Tabela 3 apresenta os resultados da simulação. É possível notar que o G-BAM com comportamento AllocTC-Sharing tem números globais melhores se comparados com o MAM e o RDM: volume de tráfego atendido maior; menor número de LSPs bloqueadas; menor número de LSPs preemptadas; e maior número de LSPs atendidas. Este resultado decorre da estratégia oportunista utilizada pelo AllocTC-Sharing, que permite compartilhamento HTL e LTH.

Os resultados das simulações demonstram que o AllocTC-Sharing consegue trabalhar mais próximo da capacidade da rede. Este comportamento levou a disponibilizar largura de banda a um número de 1.589 LSPs a mais que o RDM e 13.716 a mais que o MAM. Como consequência, o percentual médio de largura de

banda atendida, considerando todas as CTs, é cerca de 1% maior em relação ao RDM e cerca de 8% maior em relação ao MAM, conforme ilustrado na Figura 2.

Tabela 3 - Resultados gerais do primeiro cenário de simulação

	G-BAM MAM	G-BAM RDM	G-BAM Alloc
Número de LSPs Geradas	215.636		
Número de LSPs Bloqueadas	38.028	15.902	12.066
Número de LSPs Preemptadas	0	10.233	9.768
Número de LSPs Atendidas	176.984	189.111	190.700
Total de Banda Gerada	4.310.324		
Total de Banda Atendida	3.393.779	3.704.532	3.743.722

Os resultados indicam que o isolamento inflexível característico do MAM implica em redução da eficiência no uso da largura de banda. Este modelo é adequado para ambientes DS-TE em que se quer garantir apenas proteção contra degradação de QoS para algumas CTs. Conforme esperado, os resultados do RDM mostram uma flexibilidade maior no compartilhamento da banda, permitindo uma melhor eficiência no uso da capacidade do enlace, em razão da estratégia de compartilhamento HTL. Contudo, o uso exclusivo da HTL implicou num melhor atendimento ao tráfego de tempo real e/ou de uso intensivo de largura de banda (CT1) do que ao tráfego crítico e/ou de emergência (CT2). Este resultado é incompatível com a hierarquia entre CTs definida no esquema de classificação de tráfego proposto, visto que a CT2 foi projetada para abarcar as aplicações mais prioritárias do ponto de vista de requisitos de QoS.

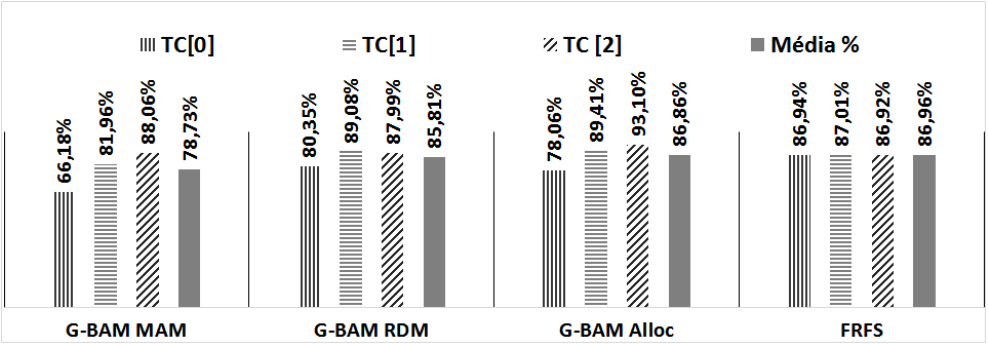


Figura 2 - Percentual de banda atendida por CT

Finalmente, os resultados do G-BAM com comportamento AllocTC-Sharing indicam um percentual maior de utilização da capacidade da rede associada a uma alocação de recursos compatível com a hierarquia definida entre as CTs, contribuindo para a funcionalidade das diversas aplicações que utilizam a rede. Estas características são implicações diretas da utilização concomitante das estratégias de compartilhamento HTL e LTH entre CTs. É um modelo mais flexível e que responde melhor à dinâmica de variações de perfis do tráfego ao longo do tempo, sendo, portanto, apropriado ao propósito de gerenciamento definido.

Resultados - Proporcionalidade

Para [BEREC 2011], uma prática é proporcional quando os meios utilizados para atingir seus objetivos não são maiores do que é apropriado e necessário para atingir esses

objetivos. Assim, busca-se demonstrar que o G-BAM com comportamento AllocTC-Sharing não interfere no tráfego mais do que o necessário para atingir seu propósito. Para este tipo de avaliação, BEREC sugere utilizar como referencial de comparação uma configuração sem o uso de Classes de Tráfego.

A configuração sem o uso de Classes de Tráfego irá colocar qualquer tráfego no mesmo nível. Assim, esta configuração, que será chamada de *First-Request-First-Served* – FRFS, "primeiro a ser requisitado, primeiro a ser servido", disponibilizará a largura de banda dos enlaces para todos os tipos de tráfego, sem prioridades, mesmo durante congestionamento. A única restrição é a disponibilidade de largura de banda no enlace no momento da requisição. A marcação do tráfego por CT continuará necessária com o objetivo estrito de comparar os resultados desta simulação com os gerados pelo AllocTC-Sharing.

Para ilustração da diferença de comportamento do Alloc e do FRFS, foi escolhido o enlace 27 (origem nó 9 e destino nó 7). Este enlace apresentou um perfil de utilização muito alto – congestionamento. Assim, é possível evidenciar as diferenças de operação entre a configuração FRFS e o comportamento AllocTC-Sharing. Conforme ilustrado na Figura 3, para o AllocTC-Sharing é possível perceber a predominância de CT2 sobre CT1 e sobre CT0. Já a configuração FRFS não impõe qualquer diferenciação no atendimento ao tráfego marcado por CT.

Os resultados estatísticos de desempenho global (Figura 2), considerando inclusive enlaces não congestionados, indicam que o AllocTC-Sharing e o FRFS alcançam praticamente o mesmo nível de utilização da capacidade da rede (diferença média de 0,1%). Contudo, o FRFS, diferentemente do AllocTC-Sharing, não contribui para o adequado atendimento do tráfego de aplicações que tem uma maior sensibilidade a requisitos de qualidade (CT2 e CT1), e, consequentemente, não contribui para a funcionalidade destas aplicações, especialmente nas situações em que os enlaces passam por momentos de congestionamento. O FRFS trataria uma chamada VoIP ou uma comunicação de emergência da mesma forma que uma mensagem de e-mail (por exemplo, fornecendo o mesmo atraso e a mesma probabilidade de atendimento na entrega de pacotes).

O FRFS seria mais adequado a redes de melhor esforço, onde nenhuma garantia de QoS é requerida. Em outras palavras, uma configuração sem classes de tráfego, apesar de ser menos interferente, não é a forma mais efetiva de se tratar um conjunto de tráfego com uma variedade tão grande de requisitos de QoS. Por outro lado, o AllocTC-Sharing, ao permitir isolamento entre as CTs durante períodos de congestionamento, promove certa proteção contra degradação de QoS para todas as CTs, além de trabalhar bem próximo da eficiência máxima dos enlaces em razão de suas estratégias de compartilhamento de largura de banda. Desta forma, como a opção menos interferente não se mostrou igualmente efetiva, o mecanismo avaliado pode ser considerado uma solução proporcional.

As características de operação do G-BAM com comportamento AllocTC-Sharing demonstradas nesta subseção comprovam que a solução alcança o propósito de gerenciamento definido, na medida em que contribui para a funcionalidade das diversas aplicações; segue estritamente o esquema de classificação de tráfego não discriminatório proposto; e é um mecanismo proporcional; atendendo

concomitantemente aos requisitos nº 1, 4 e 5 para qualificação da prática como gerenciamento de tráfego razoável.

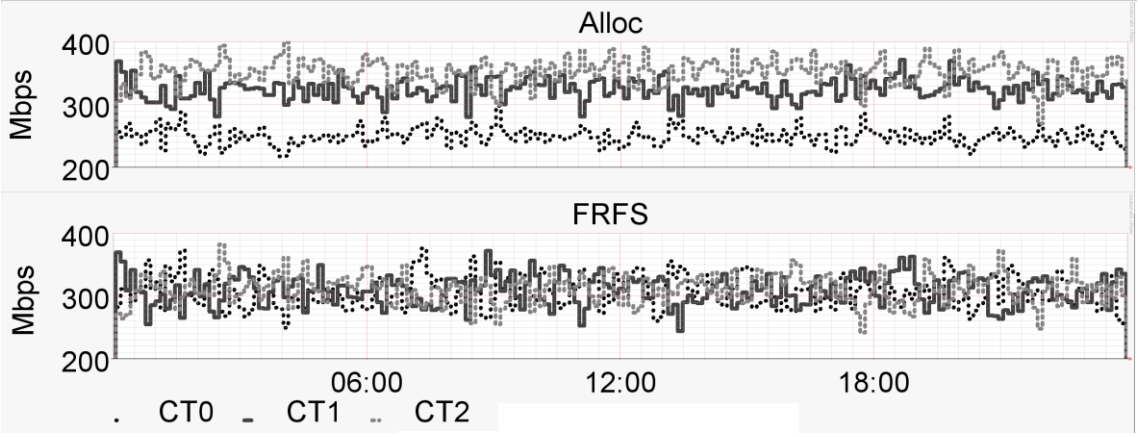


Figura 3 - Utilização do enlace 27 – G-BAM com AllocTC-Sharing vs FRFS

5.2.2. Segundo Cenário de Simulação – Configuração e resultados

Nesta subseção são apresentados os resultados que comprovam que a intervenção no tráfego que o G-BAM com comportamento AllocTC-Sharing executa, como bloqueios, devoluções e preempções de LSPs, ocorrem de forma excepcional apenas durante períodos de escassez de largura de banda, i. e., de congestionamento. Para tal, utiliza-se três diferentes perfis de ocupação da rede: subutilização, alta utilização com momentos de congestionamento e congestionamento prolongado.

Nesta simulação de 5 horas de duração, foi definido como única fonte de tráfego o nó 0 e como único destino o nó 1 (correspondentes ao enlace 0) da topologia utilizada. Desta forma, o enlace 0 experimenta os três diferentes perfis de utilização da largura de banda necessários para esta prova de conceito: (1) uma fase de subutilização do enlace, (2) uma fase de alta ocupação, com variações de tráfego entre as CTs e (3) uma fase de congestionamento prolongado. Os parâmetros configurados para cada LSP são os seguintes:

- Tempo de vida das LSPs – modelagem exponencial – média de 300 segundos;
- Carga das LSPs – Distribuição aleatória entre 5 Mbps e 15 Mbps;
- Critério de parada – 5 horas (18.000s); e
- Intervalo médio entre chegadas de solicitações de LSPs modelado exponencialmente tendo como média LSPs geradas conforme indicado na Tabela 4.

A fase 1 da demonstração representa a primeira hora da simulação e é caracterizada por baixa taxa de geração de LSPs para todas as CTs, i. e., uma baixa ocupação do enlace. As fases 2, 3 e 4 são caracterizadas por alta ocupação dos enlaces, com alguns momentos de congestionamento, levando ao bloqueio de LSPs. Na fase 2 ocorre chegada de LSPs associadas à CT0 numa taxa maior. Na fase 3 a taxa de chegada de novas LSPs associadas a CT0 permanece alta enquanto também é alta a taxa de chegada de novas LSPs associadas a CT1. Na fase 4, o fluxo de chegada de novas LSPs de CT1 é reduzido enquanto as LSPs de CT2 tem sua taxa de chegada aumentada e as de CT0 tem a taxa mantida. Durante a fase 5 tem-se alta taxa de geração de LSPs

associadas a todas as CTs. Nesta última fase, caracterizada como de congestionamento prolongado, a quantidade de bloqueios de LSPs tende a ser maior do que nas outras.

Tabela 4 – Intervalo médio (em segundos) entre chegadas de solicitações de LSPs por CT

Fase	1	2	3	4	5
Intervalo	0:00 - 1:00	1:00 - 2:00	2:00 - 3:00	3:00 - 4:00	4:00 - 5:00
CT0	50	4	8	8	4
CT1	50	50	8	50	4
CT2	50	50	50	8	4

Resultados - Excepcionalidade

Com o AllocTC-Sharing, a aplicação de prioridades (diferenciação de serviço) no estabelecimento ou manutenção de LSPs só ocorre quando o enlace está congestionado. Esta aplicação de prioridades vai resultar em preempção ou devolução de LSP estabelecida, quando não houver largura de banda suficiente no enlace para atender a nova requisição de LSP e houver saldos positivos de banda emprestada (HTL e LTH, respectivamente); e bloqueio de LSP, quando não houver largura de banda suficiente no enlace para a atender a nova requisição e não houver saldos positivos de banda emprestada.

A Figura 4 ilustra a utilização do enlace 0 durante a operação do G-BAM com comportamento AllocTC-Sharing. Na fase 1, de subutilização do enlace, o índice de bloqueios é nulo, visto que existe sobra de largura de banda para todas as CTs. Isso significa que nenhum tráfego é descartado e não é aplicada prioridade no estabelecimento ou manutenção de LSPs. Nas fases de 2 a 4, o índice de bloqueios é médio, em razão da alta ocupação dos enlaces e de alguns momentos de congestionamento.

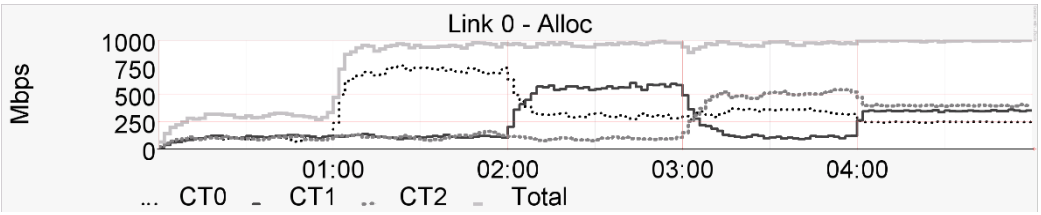


Figura 4 - Utilização do enlace 0

Na fase 5, o índice de bloqueios é alto, em decorrência da demanda por largura de banda ser maior que a disponibilidade durante todo o período, conforme ilustrado na Figura 5. De forma similar, os episódios de preempções de LSPs, característica de operação do compartilhamento HTL e os episódios de devoluções de LSPs, característica de operação do compartilhamento LTH, ambos utilizados pelo AllocTC-Sharing, vão ocorrer somente nos momentos de disputa por recursos, nunca em momentos de subutilização do enlace.

Estes resultados comprovam que a intervenção no tráfego executada pelo G-BAM com comportamento AllocTC-Sharing, na forma de aplicação de prioridades, ocorrem de forma excepcional (atendendo ao requisito nº 8), ou seja, ocorrem apenas

durante períodos de congestionamento. Consequentemente, os resultados comprovam o atendimento igualitário do tráfego, como regra.

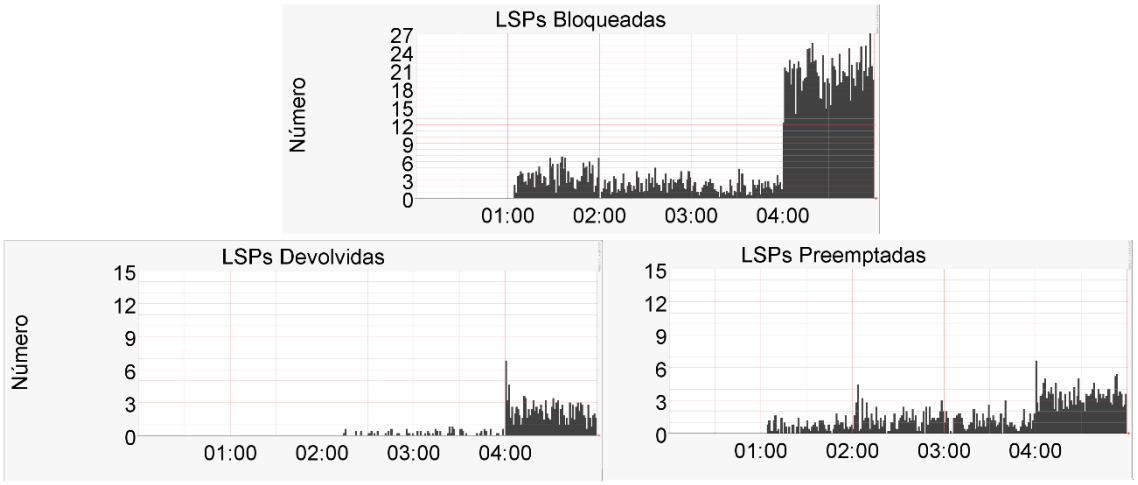


Figura 5 – Número de bloqueios, preempções e devoluções no enlace 0 – Alloc

6. Conclusões e Trabalhos Futuros

O uso do G-BAM com comportamento AllocTC-Sharing foi apresentado como um mecanismo de gerenciamento de tráfego aplicável a ISPs com o propósito de contribuir para a funcionalidade das diversas aplicações da Internet, de forma compatível com as regras de neutralidade de rede. Sua operação foi implementada e simulada com o objetivo de verificar sua conformidade com a neutralidade de rede, isto é, o atendimento tanto à regra de não discriminação quanto aos requisitos para qualificação como gerenciamento de tráfego razoável.

Em resumo, o G-BAM com comportamento AllocTC-Sharing respeita a regra de não discriminação e aplica diferenciação de serviço com o uso de Classes de Tráfego de forma razoável, maximizando a utilização da capacidade da rede e contribuindo para o atendimento dos variados requisitos de qualidade das aplicações atendidas pela rede. Desta forma, sua operação se mostra, ao mesmo tempo, compatível com as regras de neutralidade de rede impostas por diversas ARNs e eficiente do ponto de vista de utilização da largura de banda, implicando potencialmente na diminuição do impacto regulatório de tais regras.

Trabalhos futuros incluem pelo menos duas interessantes possibilidades: (1) avaliação da relação entre os índices de atendimento a requisição de LSPs (concessão de largura de banda) com índices de referência para a QoE de algumas aplicações; e (2) estudo e proposta de diretrizes para fiscalização de mecanismos de gerenciamento de tráfego em redes IP, dentro e fora do ambiente MPLS.

Referências

BEREC. A framework for Quality of Service in the scope of Net Neutrality, 2011.

BEREC. BEREC Guidelines on the Implementation by National Regulators of European Net Neutrality Rules, 2016.

Decreto nº 8.771. 11 de maio de 2016. Regulamenta a Lei nº 12.965, de 23 de abril de 2014, para tratar das hipóteses admitidas de discriminação de pacotes de dados na

internet e de degradação de tráfego. Diário Oficial [da República Federativa do Brasil], 2016. Brasília, DF, 11 mai. 2016, Edição Extra, Seção 1, p. 7.

Lei nº 12.965. de 23 de abril de 2014. Estabelece princípios, garantias, direitos e deveres para o uso da Internet no Brasil. Diário Oficial [da República Federativa do Brasil], 2014. Brasília, DF, Seção 1 - 24 abr. 2014, Pág. 1.

FCC. FCC Releases Open Internet Order, 12 de março 2015. Disponivel em: <https://apps.fcc.gov/edocs_public/attachmatch/FCC-15-24A1_Rcd.pdf>. Acesso: outubro 2016.

ITU-T. Y.1221 - Traffic control and congestion control in IP-based networks, 2010.

ITU-T. Y.1541 - Network performance objectives for IP-based services, 2011.

Jordan, S. A Layered Network Approach to Net Neutrality. *International Journal of Communication*, 2007. 427-460.

Le Faucheur, F.; Lai, W. Requirements for Support of Differentiated Services-Aware MPLS Traffic Engineering. IETF. RFC 3564. 2003.

Neto, W. D. C. P.; Martins, J. S. B. A RDM-like Bandwidth Management Algorithm for Traffic Engineering with DiffServ and MPLS Support. *International Conference on Telecommunications - ICT*. São Petersburgo, Rússia: IEEE. 2008. p. 1 - 6.

Ramos, P. H. S. Arquitetura da Rede e Regulação: A Neutralidade da Rede no Brasil. 2015. Fundação Getúlio Vargas. Dissertação (Mestrado).

Reale, R. F. et al. Uma Visão Tutorial dos Modelos de Alocação de Banda (BAM - Bandwidth Allocation Models) como Mecanismo de Provisionamento de Recursos em Redes IP/MPLS/DS-TE. *Revista de Sistemas e Computação*, Salvador, v. 5, n. 2, p. 144-155, dez 2015.

Reale, R. F.; Bezerra, R.; Martins, J. S. B. G-BAM: A Generalized Bandwidth Allocation Model for IP/MPLS/DS-TE Networks. *International Journal of Computer Information Systems and Industrial Management Applications - IJCNC*, v. 6, p. 635-643, 2014.

Reale, R. F.; Neto, W. D. C. P.; Martins, J. S. B. Modelo de Alocação de Banda com Compartilhamento Oportunista entre Classes de Tráfego e Aplicações em Redes Multiserviço. *XXIX Simpósio Brasileiro de Redes de Computadores*. Campo Grande, MS: 2011. p. 163 - 176.

Schewick, B. V. Network Neutrality and Quality of Service: What a nondiscrimination rule should look like. Forthcoming, *Stanford Law Review*, 67, n. 1, 2015.

Webb, M.; Henderson, W. Net neutrality: A regulatory perspective. *ITU Global Symposium for Regulators*, 2012.

Wójcik, R. Net Neutral Quality of Service Differentiation in Flow-Aware Networks. *Poznan University of Technology*. Kraków, Poland. 2011. Tese (Doutorado).

Wu, T. Network Neutrality, Broadband Discrimination. *Journal of Telecommunications and High Technology Law*, 2, 2003.

Join rate improvements in P2P live streaming based on topological aspects during flash crowds

Eliseu César Miguel[1], **Fernando C. S. Coelho**[2], **Bruno Morgan**[1], **Murilo Calado Junior**[1], **Ítalo Cunha**[2], **Sérgio Campos**[2]

[1]Instituto de Ciências Exatas
Universidade Federal de Alfenas (UNIFAL-MG)
Alfenas, Brasil

[2]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, Brasil

{eliseu,bruno.morgan,murilo.cesar}@bcc.unifal-mg.edu.br,
{fccoelho,cunha,scampos}@dcc.ufmg.br

Abstract. *In P2P networks peers share content in a topological overlay above the physical network. This is fundamental to network's performance. However, the simultaneous arrival of many peers, known as* flash crowd*, can affect network topology and disrupt content transmission. Current studies frequently separate flash crowd from topology arrangement techniques. In this work, we set topological partnership restrictions to preserve the existing mesh during a flash crowd. Using a* parallel network *technique, incoming peers are isolated in sub channels to avoid new relationship interference between newcomers and existing peers. This is particularly important because incoming free riders may affect existing cooperative peers significantly. In our experiments, we show that this restriction has allowed the transmission to remain unaffected in the presence of a flash crowd six times greater than would have been possible without the proposed technique.*

Resumo. *Nas redes P2P para vídeo ao vivo, os clientes compartilham o conteúdo em uma organização topológica sobreposta à rede física, fundamental para o bom desempenho das redes. Contudo, a chegada de grande número de clientes simultaneamente, o* flash crowd*, em casos extremos, pode desestruturar a topologia da rede e interromper a transmissão do conteúdo. Em geral, os trabalhos separam estudos sobre flash crowd dos estudos de topologia. Neste trabalho, configuramos restrições topológicas de parcerias para preservar a malha durante o flash crowd. Com uso da técnica de* redes paralelas*, os clientes recém chegados foram isolados em subcanais de transmissão para evitar que os clientes não cooperativos interferissem nas novas parcerias. Em nossos experimentos, mostramos que estas restrições de parcerias permitiram ingressar simultaneamente uma quantidade de clientes seis vezes maior que a quantidade ingressada sem o isolamento dos clientes não cooperativos.*

1. Introduction

Differently from the client/server content distribution model, P2P networks allow live streaming to a large audience without reliance in server's upload bandwidth. In these networks, each client needs to share the received content, what makes these systems scalable and with low running costs. Real systems allow thousands of simultaneous users in many video distribution channels [UUSee 2008, SopCast 2008, PPLive 2008, TVU 2013]. In these systems, peers establish partnerships in a decentralized way what creates a mesh topology over the physical network to exchange the media content. In this case, the content is a real time video splited in *chunks*.

Many problems may arise under the P2P network during topology maintenance and during chunks distribution. P2P networks are constantly hit by the peer's dynamism, known as *churn*, which is caused by peers leaving and joining the network. Churn may harm the systems [Zheng et al. 2011]. Furthermore, the presence of uncooperative peers, known as *free riders*, reduces the chunks availability what decreases chances of good partnerships to receive data [Adar and Huberman 2000, Moltchanov 2011, Karakaya et al. 2009, Krishnan et al. 2004, Meulpolder et al. 2012].

Many P2P strategies were proposed over the time in order to ensure a media transmission with low chunk latency and low chunk discontinuity. In this case, latency accounts for the time gap between the chunk generation on the server up to it's visualization in the peer. Basically, these strategies seek to establish good relationships among peers and at the same time to bring cooperative peers close to the media server [Payberah et al. 2011, Lobb et al. 2009, Fortuna et al. 2010]. These topology interventions may require time so that peers acquire neighborhood awareness until they establish promising partnerships.

On the other hand, flash crowd happens suddenly. So, there are techniques to control the joining rate which are capable of dealing with the huge amount of newcomers what prevents the network from disruption [Liu et al. 2009, Chen et al. 2011, Li et al. 2008, Liu et al. 2012]. Among these techniques, in [Liu et al. 2012], the flash crowd is controlled by splitting new peers in *joining batches*[1] which receives joining permission for joining up the network in time slots. Thus, these time slots allow the network to establish new partnerships. In this way, this procedure is repeated until the end of all the flash crowd event. In this technique, many peers await in queues for accessing the network. In this context, upload bandwidth is wasted while potential cooperative peers are waiting their turn to join the mesh. In cases where the waiting time is longer, peers may give up from the waiting queue, what becomes a major issue.

Another issue with flash crowd control technique is to estimate network's resources in order to know it's joining potential. Network's surplus resources are parameters to mathematical models which manages *joining batches* creation and are used to define each joining time slot. Although, as happens in [Liu et al. 2012], previous works about flash crowd focus in resources related to the idle upload bandwidth capacity and some aspects such as the partnership amount allowed to each peer, without any consideration around network's topology.

[1]Normally, batch is used for tree topology, and *slot* for mesh topology. However, we use batch in this context to distinguish from *time slot*

Differently from previous works, we fixed the bandwidth resources and peer's amount of partnerships in the network to evaluate elements related to the topology that can increase the amount of newcomers' peers in a joining batch without compromising the network's stability. We found that it is possible to change partnership criteria and create greater joining batches without any loss to cooperative or uncooperative peers. In our experiments, changes imposed to partnerships insulate newcomer free riders from cooperative peers resident in the network during the flash crowd event. This has allowed a peer joining rate six times greater in a single joining batch, without compromising the network's service.

As contributions of this work, we highlight: first, we show that it is possible to mitigate free riders negative presence effects during the flash crowd event by only setting restrictions to partnerships between free riders and cooperative peers. Furthermore, it is shown that these restrictions allow that both, cooperative peers and free riders, can join simultaneously the network without disrupting the live media transmission.

In a second place, we present a parallel network approach for handling flash crowd event. In this way, even with resources constraints, as will be discussed in this work, a large number of free riders peers are able to join the network without competition with cooperating peers resources. This scenario is possible because parallel networks were set to not allow partnership formation among free riders and cooperative peers that joined the network before the flash crowd event, what is enough for wiping out the risks offered by free riders exhausting network's resources.

Finally, we present *Free Rider Slice* as a simple way to restrict partnerships among free riders and cooperative peers without the need to implement parallel networks. We believe that free rider slice concept can be implemented in tandem with topological techniques to improve the network's live media distribution and to prepare the network overlay to receive a sudden flash crowd event.

The remainder of this work is organized as follows: Section 2 discusses related works. We describe TVPP, the real P2P streaming system that we use in our work, in Section 3. We explain our experimental method in Section 4 and present our results in Section 5. We offer conclusions in Section 6.

2. Related works

The P2P model is characterized by peers directly connecting to each other without intermediaries. The partners for a given peer can be classified into in-partners and out-partners. P2P systems may impose constraints on the number of in-partners (in-degree) and/or out-partners (out-degree). An in-depth look at the literature reveals that most of the works, such as [Traverso et al. 2015, Lobb et al. 2009, Payberah et al. 2011], do not impose fixed limits for the out-degree implicating in a non-zero chance that the number of out-partners exceed the bandwidth capacity of a given peer.

Therefore, additional techniques for choosing out-partners priority for chunk sending may be used [d. Silva et al. 2008, Wu et al. 2012]. Thus, low upload bandwidth peers can manage a large number of partners without cracking the video transmission. Unlimited out-degree may be an optional topology approach to improve overlay arrangement. However, according to [Lobb et al. 2009], large neighborhoods deliver a burden

to the network performance due to the need of each peer maintain and exchange a large amount of information with their neighbors, thus, wasting bandwidth and increasing the complexity of the scheduling algorithm.

Furthermore, overlay topology organization should dispose a distributed or centralized way to detect peers chunk upload contribution. Free rider identification is a challenge and demands time to be accomplished. *Conscious Free rider* was introduced in [e Oliveira et al. 2013]. In cases that upload bandwidth is limited, as happens with mobile peers, it is preferable that these peers join as conscious free riders. Thus, cooperative peers do not waste time requesting chunks for these uncooperative peers neither choose them for in-partnerships. On the other hand, in the flash crowd event, approaches such as in [Liu et al. 2009], ask for peer's bandwidth before bootstrapping each peer in the network. Then, peers are sorted by upload bandwidth and uncooperative peers, such as free riders, can be stalled for long time. This time is managed by *Deadline Alarm Queue* which avoids peers from giving up the joining stage. These works show that it is important to improve topology overlay organization during the flash crowd.

The majority of flash crowd handling techniques are based on peers joining by time slots. In this case, these time slots constraints allow networks to scale with the increase in the number of users, given the absence of multicast support in global network-layer [Rückert et al. 2015]. TOPT, presented in [Rückert et al. 2015], is built upon the state-of-the-art hybrid streaming approach called TRANSIT [Wichtlhuber et al. 2014] and consider to prepare the system overlay, beyond using only the time slots, for a flash crowd event. TOPT is a rare approach with this focus.

To mitigate flash crowd effects without advanced flash crowd control techniques, streaming systems such as COOLSTREAMING or PPLIVE rely on a large number of dedicated servers or the use of CDNs [Wu et al. 2012, Liu et al. 2012]. Thereby, it increases the systems cost, besides the challenges in dealing with the unforeseen flash crowd's occurrences. Differently, in this work we want to study uncooperative relationship constraints to make robust overlay with or without flash crowd event. We define a limited network bandwidth and fixed value of out-degree configurations to understand whether it is possible to mitigate flash crowd degradation without disrupting media transmission for all peer classes.

Despite other works, we believe that once defined both enough small in-degree and out-degree, the flash crowd consequences are softened, without sophisticated techniques for choosing out-partners priority. Furthermore [Liu et al. 2012] shows that neighborhood size does not boost peers' joining time. In this case, it have been compared neighborhood sizes in the range of 6-100 and found out that joining time of 20 against 100 peers neighborhood sizes, had similar joining time behavior.

Regarding free rider issues, we consider that their presence are allowed in network, but conscious free riders reduce the complexity of joining techniques and overlay management techniques. Then we chose these free riders to constrain network's resources on the experiments. In the same way we avoid any technique for choosing out-partners to send chunks and also, differently from [Liu et al. 2009], we allow continuous joining of peers in our experiments.

Moreover, we investigate whether it is possible to reach good P2P system's result,

without the need of sophisticated techniques, but through just configuring parameters such as neighborhood size. Finally, we believe that free rider's relationship constraints may help new topology organization works to obtain robust overlay meshes for facing flash crowd events.

3. P2P Live Streaming System

We define a P2P live streaming system as a system with a set of *peers* that collaborate with each other to watch a live media transmission. The *server* S is a special peer that encodes the video, splits the video into *chunks* (a chunk can contain multiple frames), and starts the transmission.

Each peer p has a set of *partners* $\mathcal{N}(p)$ for exchanging video chunks. To get more control over partnerships, $\mathcal{N}(p)$ is split between two subsets of partner peers: $\mathcal{N}_i(p)$ containing *in-partners* (partners that provide chunks to p) and $\mathcal{N}_o(p)$ containing *out-partners* (partners that receive video chunks from p).

The maximum number of in-partners is denoted by $N_i(p)$. Similarly, the maximum number of out-partners is denoted by $N_o(p)$. For S, $\mathcal{N}_i(p) = \emptyset$. In order to join a live streaming channel, a peer p registers itself at a centralized *bootstrap* server B, which returns to p a subset of all peers currently active in the system as potential partners. Peer p selects peers from this subset and tries to establish partnerships with them.

All relationship $p \in \mathcal{N}_o(p')$ requires $p' \in \mathcal{N}_i(p)$. Successfully established partnerships determine $\mathcal{N}(p)$. When p detects that one of its partners $p' \in \mathcal{N}(p)$ has been silent for longer than a predefined time period, p removes p' from $\mathcal{N}(p)$. It is not a problem because peer p periodically contacts the bootstrap server to obtain a new list of potential partners to replace the lost partnership.

Several works do not impose a fixed value for N_o of peers in the network. It ensures that if p is chosen by p' for $\mathcal{N}_i(p')$ implies that p should accept p' in $\mathcal{N}_o(p)$ [Lobb et al. 2009, Traverso et al. 2015]. On the other hand, we have imposed known values for both, N_i and N_o, while peers randomly choose in-partners. To organize the network topology, peers are able to accept a new out-partner even when $\mathcal{N}_o$ is full. In this case, p disconnects a random out-partner $q \in \mathcal{N}_o(p)$ with less out-partnerships than the new peer n, i.e., $N_o(q) < N_o(n)$. Afterwards, peer p remains unable to disconnect more out-partners to accept incoming partnership requests during the next 60 seconds to prevent overlay instability.

Each peer has a local buffer to store its video chunks. Periodically, peers exchange *buffer maps* with their partners $\mathcal{N}_o$ to inform them which chunks they have available. Each peer periodically checks which chunks it needs, identifies which partners $\mathcal{N}_i(p)$ can provide missing chunks, and then sends chunks requests accordingly. In this work, we schedule chunk requests using the earliest deadline first policy associated with *Simple Unanswered Request Eliminator* (SURE) [e Oliveira et al. 2013]. SURE allows each peer p to hold a black list containing non responding peers to avoid new requests faults. We do not limit the number of simultaneous (pending) requests. However, we limit the number of request retries to six to control each peer's opportunities to download a chunk and make download opportunities independent of buffer size. A peer considers that a request has timed out if it is not answered within 500 milliseconds. Finally, cooperative peers immediately serve their received requests in the order of their arrival.

Peers send their monitoring reports to the bootstrap server every 10 seconds. These reports, actually, include the number of chunks generated (only reported by the video server), sent, received, and the ones that missed their playback deadline; the number of requests sent, answered, and retried; the average forwarding path length, retry count, and time of arrival of received chunks; neighborhood size; and the number of duplicate chunks received. Thus, we use these peer reports to compute the performance metrics being evaluated: chunk latency and the chunk playback deadline miss rate.

4. Experimental Method

Our evaluation relies on real experiments that we have conducted on PlanetLab [PlanetLab 2009], using the P2P system, TVPP [Oliveira et al. 2013], and with five repetitions for each. We configured video and bootstrap servers in our university's network and used about 108 PlanetLab nodes as peers. The video server streams a 420 kbps video (about 40 chunks per second). Even though PlanetLab nodes have their own bandwidth and CPU restrictions, we impose additional upload bandwidth constraints on each peer to archive a restricted network in order to approach a realistic scenario.

Our experiments were performed in 1350 seconds. Firstly, the first 70 seconds initialize bootstrap and media servers. Then we construct a *first network*[2] with a group of 108 peers to support the suddenly arrival of 1080 new peer group from flash crowd that happens 350 seconds after each experiment beginning. In a total, we run around 1100 peer instances (i.e. 11 instances in each PlanetLab's node). To preserve packet loss and delay in a realistic scenario we do not allow peers' communication within the same PlanetLab's node.

To assess topological aspects we fixed some peers resource parameters. We are interested in understanding whether any free rider partnership restriction better improves flash crowd joining rate peers through all experiments with the same bandwidth and $\mathcal{N}(p)$ size network configurations. We set four types of upload bandwidth and partnership size for each peer. Table 1 shows peer's configuration. Since $N_o = 0$ for *class 0*, it defines a *conscious* free rider that informs to no one their *buffer maps*. Thereby, all free rider's partners know that they are unwilling or unable to upload data [e Oliveira et al. 2013]. Finally, all join peer groups have the same resources configuration.

Table 1. Network Peers Configuration

Peer Classes	Mb/s	Proportion	N_i	N_o
class 0	0.0	40%	10	00
class 1	1.5	27%	10	09
class 2	2.5	22%	10	20
class 3	4.0	11%	10	23

In order to study the flash crowd in this paper, we define: (i) *master network*, composed by the set of peers that joined on P2P network before the flash crowd's event; and (ii) *whole network* composed by the set of all active peers. Thus, before the flash crowd, *master network* and *whole network* are the same. Finally, we have a unique server media with $\mathcal{N}_o(s) = 20$ and no upload bandwidth constraint.

[2]In this work, the first network is called *master network*

5. Evaluation

In this section we evaluate the impact of free rider partnership limitation during the flash crowd event on P2P live streaming. We compare two scenarios, with and without free rider isolation, and show joined amount variation for the same network configuration. First, we present results for *common flash crowd technique* approach introduced in [Liu et al. 2012] whose results are previous benchmarks for this work. After it, we evaluate a parallel network to impose free riders' partnership limits and show that it is possible to allow larger simultaneous number of peers joining the network without disrupting the media distribution. Parallel networks are our first attempt to assess that free rider isolation can be another way to improve the amount of joined peers in P2P live streaming networks. Finally, despite parallel networks issues we present *Free Rider Slice*: an easy way to setup the network in order to mitigate free rider negative impact on the mesh.

5.1. Common Flash Crowd Technique

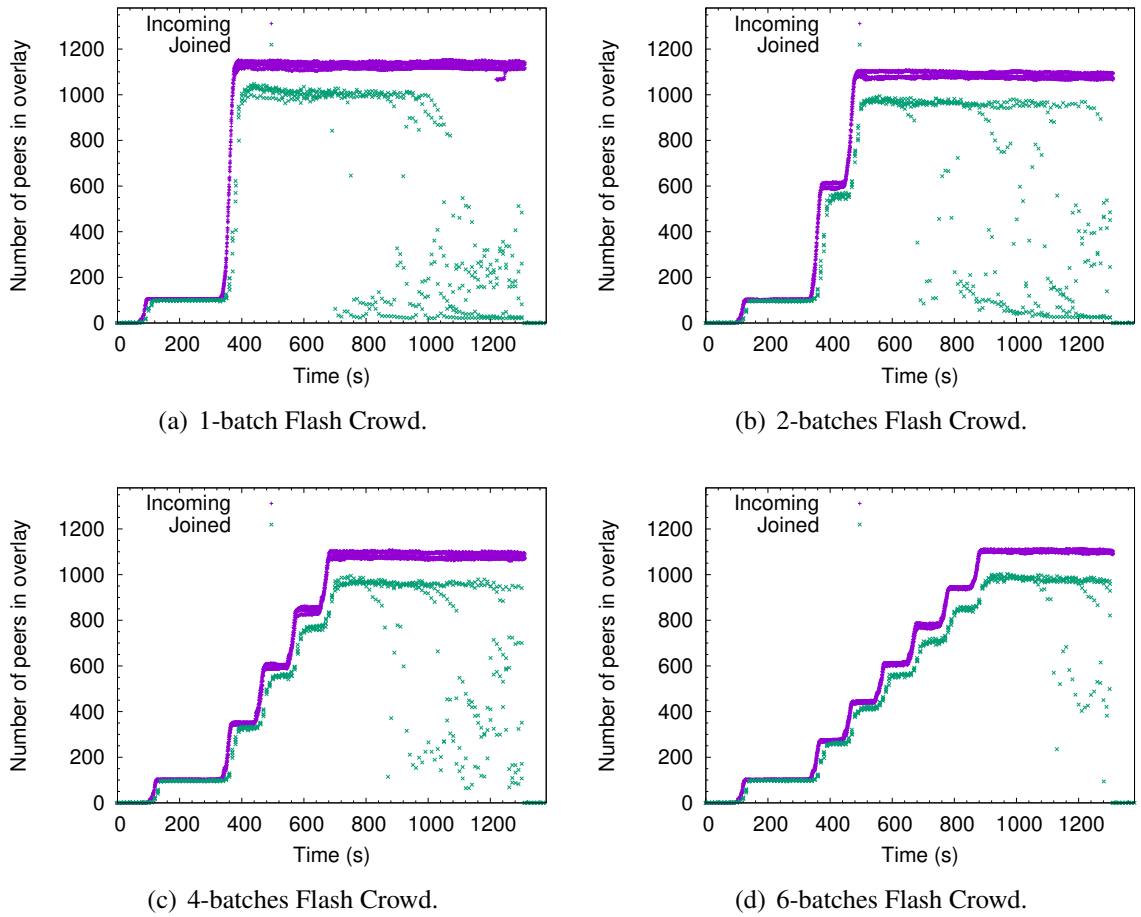
To evaluate the behavior of our network configurations we implemented and executed a flash crowd technique approach presented in [Liu et al. 2012]. Based on peer's joining by batches, this technique holds newcomers peers in a larger group $\mathcal{P}$ to control the joining process. Basically, in each i-iteration, the system evaluates the network's resources and establishes both $\mathcal{R}_i \subseteq \mathcal{P}$ and τ_i. It means that each peer $p \in \mathcal{R}_i$ is allowed to join the network and the next $(i+1)$-iterator happens after network stabilization time τ_i. Several iterations are repeated until $\mathcal{P} = \emptyset$.

To simplify understanding our network resources, we fixed $\tau_i = 100s$ and we experimented [1,2,4,6] as i-iterations limits. It is enough to take a preliminary network behavior without spending a long time implementing the common flash crowd technique in its essence since it is not the scope of our work.

Figure 1 shows the joining results charts for the common flash crowd technique. The instability in the joined amount of peers is shown in all charts on Figure 1. This demonstrates the negative effect of the flash crowd event on peer joining. Thus, we can see that no configuration had provided the desired joining amount. However, we consider the relative improvement of each iteration enough to study this hard network configuration scenario.

Once Figure 1(d) approaches the desired network joining rate, we chose this configuration to show the values of discontinuity and latency metrics for chunks in Figure 2. We observe in both Figure 2(a) and Figure 2(b) that until 1000 seconds, the network was stable. After it, since joining was not sustained, the network service was compromised and has reached high discontinuity and latency.

Our expectation is to reach at least the *6-batch* joining rate shown in Figure 1(d), but with the *1-batch* incoming peer behavior from Figure 1(a). For it, we dispose that parallel networks technique that holds newcomers in isolated correlated networks. We believe that is possible to join peers keeping distance from master network and, thus, ensuring existing peers relationships. Moreover, we expect to successfully increase peer's joining rate, by limiting the relationship amount among free riders and cooperative peers.



(a) 1-batch Flash Crowd.

(b) 2-batches Flash Crowd.

(c) 4-batches Flash Crowd.

(d) 6-batches Flash Crowd.

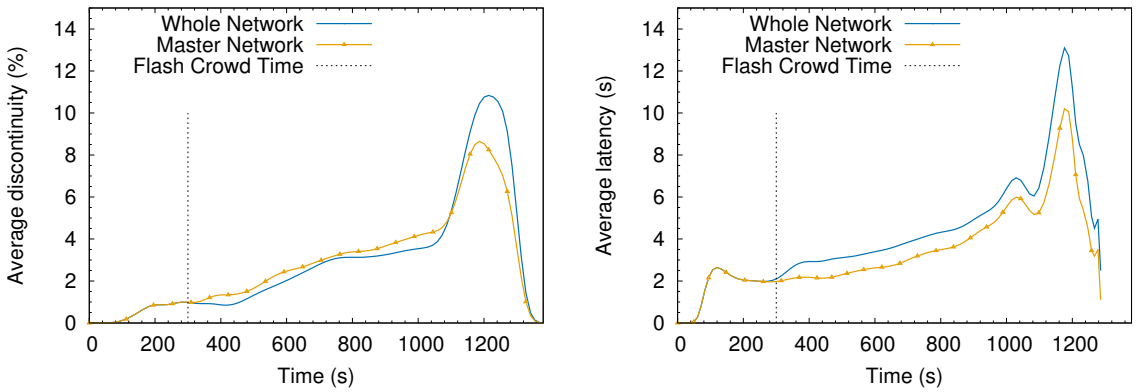
Figure 1. Common Technique Overlay Size.

5.2. Parallel Networks Technique

Parallel networks technique, introduced in [Miguel et al. 2016], are useful to construct new networks from an existent network. Let M be the *master network*, as in Section 4, the constructed P2P mesh overlay around media server S. In this technique, the bootstrap server B select a set of peers $S_{aux} \subset M$. Each $p \in S_{aux}$ becomes a especial *auxiliary server* peer. The master network ensures media transmission for auxiliary servers, but auxiliary servers are unable to give chunks for peers in the master network. This stage is called *server isolation*.

The server isolation prepares network for the flash crowd event. When the peers' herd appears, peers are distributed around each auxiliary server, establishing the parallel networks. New relationships are established only among peers in the same parallel network and their auxiliary server. It provides a first relationship limit controlled by bootstrap server. The last step is *parallel network merging*. We permit, in this step, that peers from parallel merging networks to establish new relationships among cooperatives peers from the master network. As a second limit, during merging phase we kept free riders around old partners. Thus, the merging state of parallel networks preserves the master network to sustain all the joined peers.

Figure 3 presents the stages until parallel networks formation. In the first stage, the



(a) 6-batches Playback Quality. (b) 6-batches Distribution Latency.

Figure 2. 6-batches Common Technique Metrics.

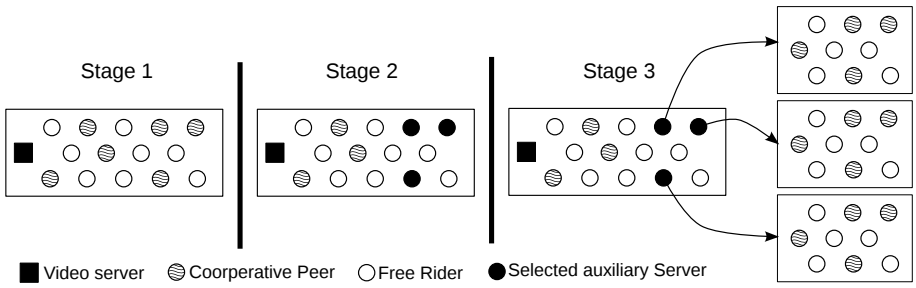


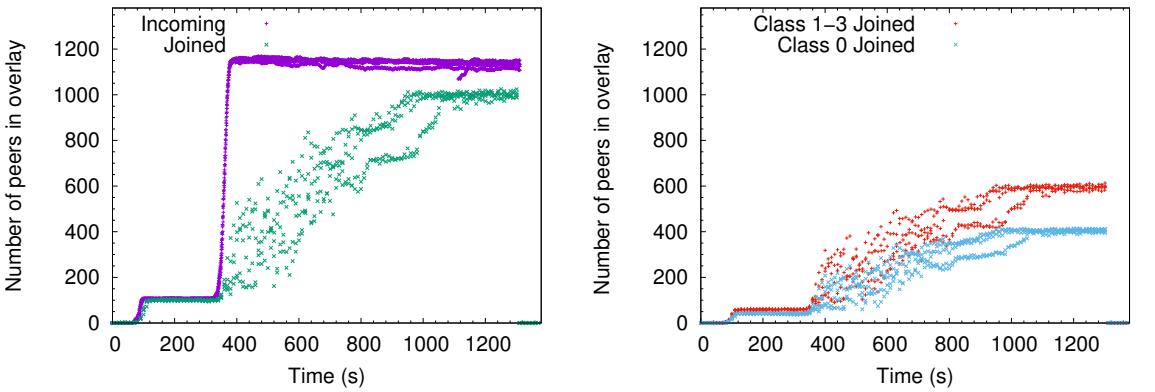
Figure 3. Parallel Networks Formation Flow.

master network is stable with the video server serving content to the peers, as described earlier in this section; in the second stage there is a flash crowd surge and the auxiliary servers are selected to attend the demand; finally, on the third stage, there is the formation of the parallel networks where selected auxiliary servers behaves as the video server for each newly formed network.

Six auxiliary server from peer *class 3* were configured in our experiments. The newcomers peers were split for constructing balanced parallel networks. After the flash crowd, we wait 200 seconds to construct the parallel network topology and then, to begin merging each of these networks in steps happening in 100 seconds interval. So, from the beginning of the experiment, since there are six parallel networks and the flash crowd event happens at 350 seconds, the first network merges at 550 seconds, the second at 650 seconds and so on.

As result, in Figure 4(a) the incoming curve is equivalent to the *1-batch* from Figure 1(a) and we have reached the same joining behavior of the *6-batches* common flash crowd technique, as shown in the joined curve in Figure 1(d). Figure 4(b) shows separated joined common peers and joined free rider. We observe balance in joining rate for all peer classes. The small joining gap between class 1-3 and class 0 is explained because free riders are 40% versus 60% from common peers.

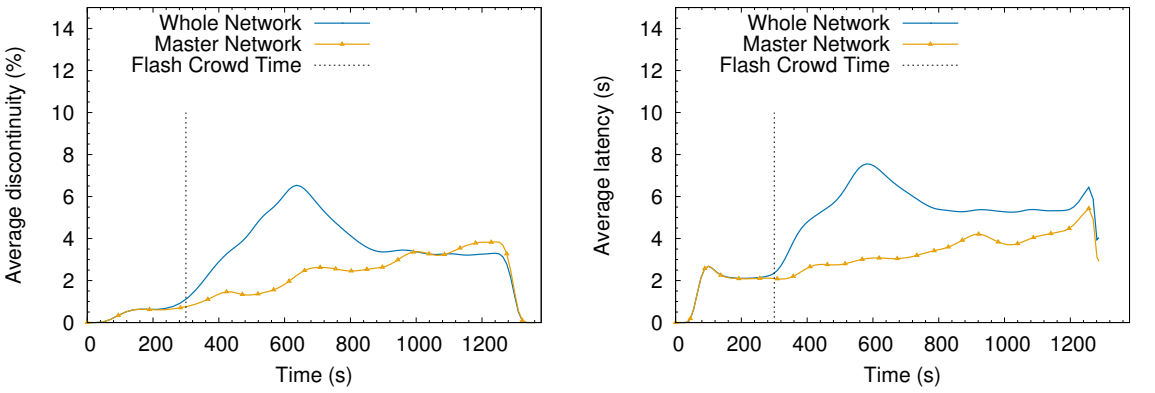
Figure 5 shows parallel networks' discontinuity and latency. It is clear that the master network was stable during the whole experiment's steps. This fact is important



(a) Overlay Size. (b) Overlay Size of Free Rider and Common Peers.

Figure 4. Parallel Networks Technique Overlay Size.

and we suspect that it explains why parallel network technique supported large number of incoming peers without postponing the joining event. In the case of the whole network, chunk latency and discontinuity peaks at around 600 seconds as expected. The poor auxiliary server upload bandwidth is a serious parallel networks constraint. Compared with non bandwidth limited server S in the master network, parallel networks received twice more incoming peers. Even so, the peaks were controlled and they stabilized during the merging steps.



(a) Playback Quality. (b) Distribution Latency.

Figure 5. Parallel Networks Metrics.

Parallel networks technique is a novel technique. Thus, we suspected that these systems work better with high bandwidth auxiliary servers. However, isolating high bandwidth peers can disrupt promising topological partnerships which may compromise the master network. Besides that, it is required a deep understanding of the stabilization time, merging phases and amount of parallel networks before they can be applied commercially. Furthermore, we issue that auxiliary server clusters and the newcomers amount in each parallel network is a challenge. Finally, we consider it was challenging to implement this technique.

Even with the lack of knowledge about these parallel networks techniques, free

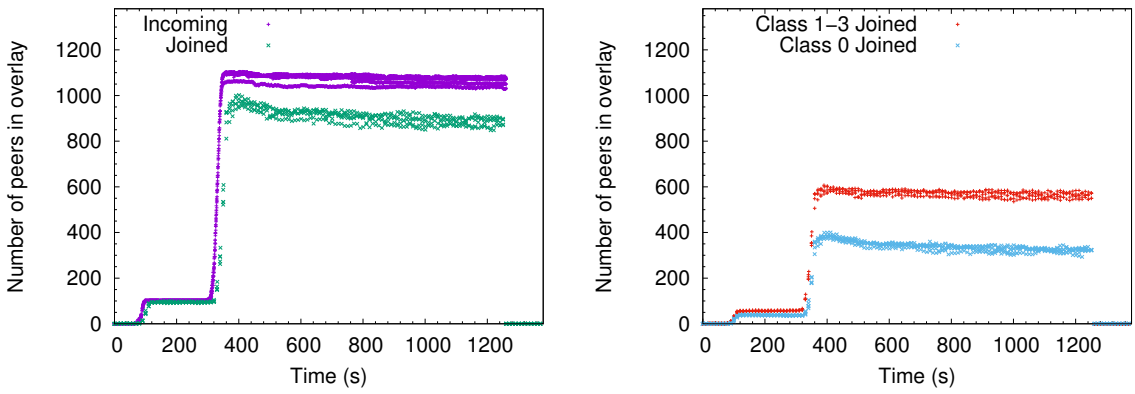
rider relationship limitation was successfully investigated in our experiments. We want, from now on, to make new topology configurations in order to explore free riders' partnership limitations in a realistic scenario. If this experiments become successful, new topology organization may contemplate this solution without service disruption and should be robust for the flash crowd events.

5.3. Free Rider Slice Setup

Based on results from parallel networks in Section 5.2, we have realized that splitting free riders from cooperative peers delivers good network performance. Furthermore, parallel networks technique requires complex implementation and is insufficiently researched yet. Thus, our goal is to decrease competition among free rider and cooperative peers imposing a simple partnership constraint on peers. Therefore, by considering peer's classification, we can define which partnerships are allowed and which are denied among peers.

Then, the free rider slice is defined following the partnership constraint: peers from class 3, that have high upload bandwidth, are allowed to accept out-partners requests only from cooperative peers, that is, peers classified on class 1-3, (i.e. classes 1, 2 and 3). On the other hand, low upload bandwidth peers from class 1 accept out-partners requests from class 0 peers (i.e. free riders). Finally, to balance the network relationship distribution, no rule was set up for peers from class 2.

As with parallel networks, we evaluate experiments considering *1-batch* incoming peers. Figure 6(a) shows joined peers amount versus incoming peers on network configured with free rider slice. We observe fast joining effect preserved until the experiment ends. Joined free riders and common peers are shown in Figure 6(b). In this case, we consider that all peers (from all classes) joined without preference nor privilege. Once server S do not accept free riders and we reach full class 0 joining, we understand that our partnership constraint rules direct new relationships among peers pushing free riders to the edge of the network topology. We believe that overlay topology is desired, once free rider and low upload peers amount from class 1 are sufficient to disrupt cooperative peers media propagation, as occurred in Section 5.1.

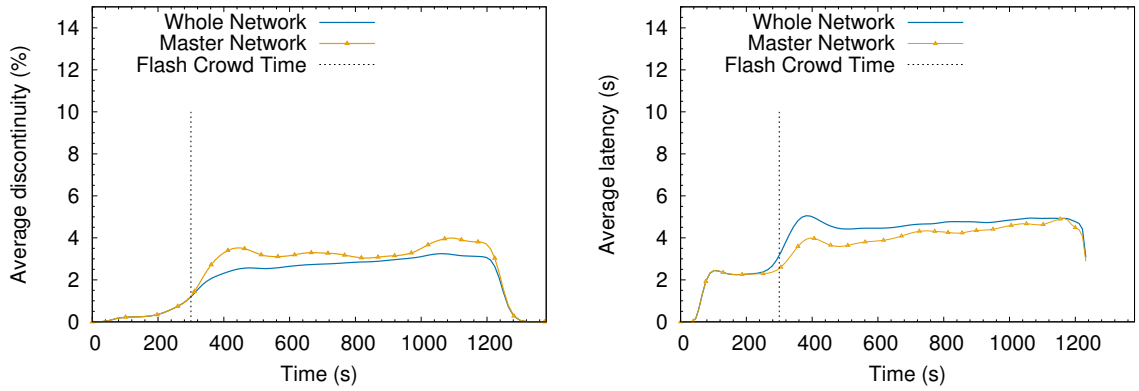


(a) Overlay Size. (b) Overlay Size of Free Rider and Common Peers.

Figure 6. Free Rider Slice Technique Overlay Size.

New topology organization improvements are observed on chunk latency and chunk discontinuity metrics, Figure 7. Both metrics stayed stable with low values for

both master and whole networks. In this case, we reach the same values of metrics observed on master network for parallel network technique, Figure 5. However, we adjust the high values of whole network's metrics found with parallel network experiments with free rider slice without large parallel network complexity. We also observe that free rider slice provides a average discontinuity below 4% throughout the experiment, which is a fair value for discontinuity, as considered by [Traverso et al. 2012].



(a) Playback Quality.

(b) Distribution Latency.

Figure 7. Free Rider Slice Metrics.

We consider that free rider slices needs more studies before being used in commercial applications. It is because the relationship among network set up parameters and its consequences are not well understood to allow dynamic classes configuration yet. Among these parameters we can mention: (i) peers' chunk distribution; (ii) peers' out-degree; and (iii) peers' upload bandwidth. However, we believe that to aggregate the concepts of free rider slice to topology organization is not complicated. This concept can inspire new future works.

6. Conclusion

Currently, there are no detailed studies about free rider relationship isolation done yet. In this work, we expose constraint for free rider neighborhood to assess whether it is a relevant issue to improve overlay organization advances. First, based on *parallel networks*, we show that free rider isolation allows growing the number of simultaneous peers joining without crashing the network transmission. Thus, we propose a new setup, called *free rider slice*, for networks' parameters set up that results a better peer joining without the *parallel networks'* complexity.

Despite other works that only identify free riders and converge the topology to push them to the mesh's edges, in this work we show that splitting the free riders and the cooperative peers' relationships may avoid media resources competition. Thus, this resource preservation contributes to the network media transmission stability. Furthermore, assigning a limited neighborhood size for each peer can become a parameter for new topology overlay techniques to avoid the negative consequences arising from non limited size.

We believe that many future works may be developed from our observations. Our experiments facing flash crowd events are a clue that simple network settings can pro-

mote major improvements in P2P live media distribution. Then, we want to study the dynamic network mechanism in order to facilitate the free rider isolation to provide a robust network topology organization.

Acknowledgment

This work was partially funded by FAPEMIG, CAPES, and CNPq.

References

Adar, E. and Huberman, B. (2000). Free Riding on Gnutella. *First Monday*, 5(10-2).

Chen, Y., Zhang, B., and Chen, C. (2011). Modeling and Performance Analysis of P2P Live Streaming Systems under Flash Crowds. In *ICC'11*, pages 1–5.

d. Silva, A. P. C., Leonardi, E., Mellia, M., and Meo, M. (2008). A Bandwidth-Aware Scheduling Strategy for P2P-TV Systems. In *2008 Eighth International Conference on Peer-to-Peer Computing*, pages 279–288.

e Oliveira, J. F., Cunha, Í., Miguel, E. C., Rocha, M. V., Vieira, A. B., and Campos, S. V. (2013). Can Peer-to-Peer Live Streaming Systems Coexist With Free Riders? In *IEEE P2P 2013 Proceedings*, pages 1–5. IEEE.

Fortuna, R., Leonardi, E., Mellia, M., Meo, M., and Traverso, S. (2010). QoE in Pull Based P2P-TV Systems: Overlay Topology Design Tradeoffs. In *P2P '10: IEEE International Conference Peer-to-Peer Computing*, pages 1–10, Washington. IEEE Computer Society.

Karakaya, M., Korpeoglu, I., and Ulusoy, O. (2009). Free Riding in Peer-to-Peer Networks. *IEEE Internet Computing*, 13(2):92–98.

Krishnan, R., Smith, M., Tang, Z., and Telang, R. (2004). The Impact of Free-Riding on Peer-to-Peer Networks. In *Annual Hawaii International Conference on System Sciences*.

Li, B., Keung, G., Xie, S., Liu, F., Sun, Y., and Yin, H. (2008). An Empirical Study of Flash Crowd Dynamics in a P2P-Based Live Video Streaming System. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–5.

Liu, F., Li, B., Zhong, L., Li, B., Jin, H., and Liao, X. (2012). Flash Crowd in P2P Live Streaming Systems: Fundamental Characteristics and Design Implications. *Parallel and Distributed Systems, IEEE Transactions on*, 23(7):1227–1239.

Liu, F., Li, B., Zhong, L., Li, B., and Niu, D. (2009). How P2P Live Streaming Systems Scale over Time Under a Flash Crowd? In *Proceedings of the 8th International Conference on Peer-to-peer Systems*, IPTPS'09, pages 5–5, Berkeley, CA, USA. USENIX Association.

Lobb, R. J., Couto da Silva, A. P., Leonardi, E., Mellia, M., and Meo, M. (2009). Adaptive Overlay Topology for Mesh-based P2P-TV Systems. In *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '09, pages 31–36, New York, NY, USA. ACM.

Meulpolder, M., Meester, L., and Epema, D. (2012). The Problem of Upload Competition in Peer-to-Peer Systems With Incentive Mechanisms. *Concurrency and Computation: Practice and Experience*, 25(7):899–917.

Miguel, E., Cunha, I., and Campos, S. (2016). Ingressos em Redes P2P para Vídeo ao Vivo. In *SBRC 2016 - WP2P+ ()*.

Moltchanov, D. (2011). Service Quality in P2P Streaming Systems. *Computer Science Review*, 5(4):319–340.

Oliveira, J., Viana, R., Vieira, A. B., Rocha, M., and Campos, S. (2013). TVPP: A Research Oriented P2P Live Streaming System. In *SBRC 2013 - Salão de Ferramentas*.

Payberah, A. H., Dowling, J., and Haridi, S. (2011). GLive: The Gradient Overlay as a Market Maker for Mesh-Based P2P Live Streaming. In *Parallel and Distributed Computing (ISPDC), 2011 10th International Symposium on*, pages 153–162.

PlanetLab (2009). An Open Platform for Developing, Deploying, and Accessing Planetary-Scale Services. `http://www.planet-lab.org/`.

PPLive (2008). `http://www.pplive.com`.

Rückert, J., Richerzhagen, B., Lidanski, E., Steinmetz, R., and Hausheer, D. (2015). TOPT: Supporting Flash Frowd Events in Hybrid Overlay-Based Live Streaming. In *2015 IFIP Networking Conference (IFIP Networking)*, pages 1–9.

SopCast (2008). Deliver Your Media to the World! `http://www.sopcast.com`.

Traverso, S., Abeni, L., Birke, R., Kiraly, C., Leonardi, E., Cigno, R. L., and Mellia, M. (2012). Experimental comparison of neighborhood filtering strategies in unstructured P2P-TV systems. In *2012 IEEE 12th International Conference on Peer-to-Peer Computing (P2P)*, pages 13–24.

Traverso, S., Abeni, L., Birke, R., Kiraly, C., Leonardi, E., Lo Cigno, R., and Mellia, M. (2015). Neighborhood Filtering Strategies for Overlay Construction in P2P-TV Systems: Design and Experimental Comparison. *IEEE/ACM Transactions on Networking (TON)*, 23(3):741–754.

TVU (2013). TVU VR. `http://www.pplive.com`.

UUSee (2008). UUSee Inc. `http://www.uusee.com/`.

Wichtlhuber, M., Richerzhagen, B., Rückert, J., and Hausheer, D. (2014). TRANSIT: Supporting Transitions in Peer-to-Peer Live Video Streaming. In *2014 IFIP Networking Conference*, pages 1–9.

Wu, H., Liu, J., Jiang, H., Sun, Y., Li, J., and Li, Z. (2012). Bandwidth-Aware Peer Selection for P2P Live Streaming Systems under Flash Crowds. In *Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International*, pages 360–367.

Zheng, Q., Long, Y., Qin, T., and Yang, L. (2011). Lifetime Characteristics Measurement of a P2P Streaming System: Focusing on Snapshots of the Overlay. In *Intelligent Control and Automation (WCICA), 2011 9th World Congress on*, pages 805–810.

Trilha Principal do SBRC 2017
Sessão Técnica 18
Redes Celulares

FuzSy: Um Escalonador baseado em Qualidade de Serviço e Lógica Fuzzy

Fabrício R. de Souza[1], **Fátima Duarte-Figueiredo**[1]

[1]Departamento de informática - Pontifícia Universidade Católica de Minas Gerais (PUC Minas)

fabricio127@gmail.com, fatimafig@pucminas.br

Abstract. *In this paper, FuzSy, or Fuzzy Scheduler, is presented. It is a cellular network packet scheduler. It is based on quality of service and fuzzy logic. Classic solutions utilize static packet scheduling, following a prearranged priority order. Static scheduling can be unfair, or waste network resources, trying to allocate resources. FuzSy utilizes metadata from the network and, from choices made by the fuzzy logic mechanism, which seeks to prioritize classes that have a higher risk of not reaching it's QoS requirements, sets a dynamic packet priority. This priority is said to be dynamic because, is one class is in a QoS position that is not favorable, the scheduler set this class' priority to be higher, regardless of any previous set priority. Fourth generation, 4G, network simulations were conducted. The results show that FuzSy was able to keep all QoS within 3GPP stipulated values and was able to achieve higher fairness, not benefiting any class at other's cost. One of FuzSy's advantages, in relation to the other schedulers, cited in this paper, is that parameters can be added to or deleted from the fuzzy mechanism. With that, on specific situations such as places hosting large scale events the scheduler can be altered and utilize different parameters and rules to better serve the users.*

Resumo. *Neste trabalho, é apresentado o FuzSy, ou Fuzzy Scheduler, um escalonador de pacotes para redes de celular. Ele é baseado em qualidade de serviço e lógica fuzzy. Soluções clássicas utilizam escalonamento estático de pacotes, seguindo prioridades previamente negociadas. O escalonamento estático pode ser injusto ou, até mesmo, pode desperdiçar recursos da rede. O FuzSy utiliza metadados da rede e, a partir da escolha feita pelo mecanismo de lógica fuzzy, que visa dar maior prioridade à classes que correm mais risco de não atingirem seus requisitos de QoS, atribui uma prioridade dinâmica aos pacotes. Essa prioridade é dinâmica porque, se uma classe se encontra em condições de QoS desfavoráveis, o escalonador passa a dar maior prioridade a ela, independentemente de qualquer atribuição feita anteriormente. Simulações de uma rede de quarta geração, 4G, foram realizadas. Os resultados mostraram que o FuzSy conseguiu manter todas as métricas de QoS dentro dos limites estipulados pelo 3GPP e conseguiu ser mais justo, não beneficiando nenhuma classe em detrimento de outras. Uma vantagem do FuzSy, em relação aos demais escalonadores, citados no trabalho, é que podem ser adicionados ou excluídos novos parâmetros e regras no mecanismo de lógica fuzzy. Com isso, situações específicas como locais com eventos de larga escala podem precisar de escalonadores com regras adaptáveis para atender melhor aos usuários.*

1. Introdução

Nas últimas décadas, a quantidade de usuários das redes de celular vem crescendo vertiginosamente e a tendência é que esse ritmo de crescimento continue. De acordo com dados disponibilizados pela Anatel (Agência Nacional de Telecomunicações) [IBGE 2016], em 2016, a taxa de dispositivos móveis por habitante no Brasil era de 1,26, o que demonstra a necessidade de acesso de qualidade a essas redes.

As redes LTE, implantadas e em uso em todo o mundo, também chamadas de redes 4G (de quarta geração), são redes de celulares que permitem acesso a diversos tipos de serviços, como, por exemplo, transmissão de voz, navegação na web, transmissão de conteúdo multimídia em tempo real, entre outros. No mundo atual, onde a quantidade de usuários simultâneos dessas redes é muito grande, é necessário que a transmissão dos pacotes seja controlada dinamicamente, para que seja possível garantir qualidade de serviço nas redes.

Numa rede com serviços diferentes, é necessário que cada um destes serviços seja tratado de forma a garantir que seus requisitos de QoS específicos sejam atendidos. Certos tipos de serviço são mais sensíveis ao atraso dos pacotes, enquanto outros dependem mais da vazão disponível, o que significa que tratar os serviços de forma igual não é interessante, visto que eles têm requisitos diferentes. Uma das maneiras de medir se uma classe de serviço foi, ou não, atendida é através da métrica de equidade. A equidade mede o quão bem um requisito de QoS foi atendido. Através do cálculo da equidade, verifica-se se as classes mais prioritárias da rede estão sendo bem atendidas, sem prejudicar as menos prioritárias, refletindo critérios justos de priorização de acesso e escalonamento na rede.

Para garantir a qualidade de serviço e a equidade entre as classes, existem mecanismos para ajustar o nível de envios de pacotes da rede. Um exemplo desses mecanismos é um escalonador de pacotes, que é responsável por decidir como e quando os pacotes da rede serão enviados. Neste trabalho, é apresentado o FuzSy (*Fuzzy Scheduler*) um escalonador de pacotes que se baseia em métricas de qualidade de serviço e um mecanismo de lógica *fuzzy* para escalonar os pacotes. O FuzSy foi proposto em uma dissertação de mestrado [de Souza 2016]. Normalmente, um escalonador utiliza uma fórmula matemática para o cálculo da prioridade de cada um dos pacotes. Diferentemente, o FuzSy utiliza um mecanismo inteligente baseado no estado atual da rede para decidir qual pacote é o mais prioritário em um determinado momento. Prioridades estáticas são prioridades que são definidas uma vez e nunca modificadas durante o envio dos pacotes. O FuzSy utiliza o conceito de prioridades dinâmicas, que se alteram, à medida que é necessário, durante o envio dos pacotes.

O FuzSy atingiu seu objetivo de prover maior equidade para as classes, conforme mostram resultados de simulações. Para todas as classes, o FuzSy foi o escalonador que apresentou maior equidade e na maioria dos casos foi o escalonador que atingiu melhores resultados para os parâmetros mais sensíveis de cada classe, quando comparado a outras propostas. O restante deste trabalho está dividido da seguinte maneira: sessão 2 elabora sobre os escalonadores de pacote existentes na literatura; a sessão 3 descreve o escalonador FuzSy; a sessão 4 apresenta os resultados obtidos através de simulações; a sessão 5 contém as conclusões do trabalho.

2. Escalonadores de pacote

Em qualquer rede de telefonia celular, na estação rádio base, existe uma fila com os pacotes que devem ser enviados para os usuários. Essa fila consiste em pacotes de diversas classes de serviço. É papel do escalonador de pacotes estabelecer a prioridade de cada pacote e reorganizar a fila de acordo essa prioridade. É necessário que o processo de escalonamento seja feito para que a qualidade de serviço da rede seja mantida, ou seja, para que uma classe não fique sem atendimento adequado. A maioria dos escalonadores de pacotes existentes na literatura utilizam fórmulas matemáticas para o cálculo das prioridades. Nesta seção são citados alguns dos escalonadores existentes na literatura.

Em [Jalali et al. 2000] o escalonador *Proportional Fair* é definido. Ele tenta maximizar a vazão da rede, não se preocupando com qualquer outro parâmetro, e utiliza a Equação 1 para calcular sua métrica.

$$P = \frac{T^{\alpha}}{R^{\beta}} \tag{1}$$

Na equação, T é a taxa de transferência disponível na antena que enviou o pacote naquele momento e R é a média da taxa de transferência daquela antena até o momento. Os valores de α e β são utilizados para regular a execução do algoritmo. Se o valor de α for escolhido como 0 e o de β como 1, o algoritmo executará como *roundrobin*, enviando todos os pacotes na ordem que eles chegarem, desconsiderando a qualidade dos canais. Se os valores forem invertidos, α como 1 e β como 0, o algoritmo sempre enviará os pacotes com a melhor qualidade de canal, ou seja, a vazão da rede será maximizada.

O escalonador FLS (*Frame Level Scheduler*) [Piro et al. 2011b] concentra-se no envio de pacotes de tempo real, calculando a quantidade necessária de recursos que deve ser alocada para que todos os pacotes de tempo real sejam enviados dentro dos seus requisitos de atraso, independentemente da quantidade de pacotes de outras classes que existam na fila de envio. Os escalonadores EXP-Rule (*Exponential Rule*) [Shakkottai and Stolyar 2002], LOG-Rule (*Loharithm Rule*) [Sadiq et al. 2011], EXP (*Exponential Proportional Fair*) [Rhee et al. 2004] e MLWDF (Modified Largest Weighted Delay First) [Andrews et al. 2001] tem o intuito de melhorar o envio dos pacotes de tempo real, adicionando um peso à métrica calculada pelo *Proportional fair*, o peso calculado por cada um está disponível na Tabela 1.

Tabela 1. Pesos dos escalonadores

Escalonador	Peso
EXP-Rule	$\sigma = exp(\frac{\sqrt[6]{\text{Atraso desejado}}^{Hol}}{1+\sqrt{\text{Média dos Hol}}})$
LOG-Rule	$\sigma = log(1.1 + (\frac{5Hol}{\text{Atraso desejado}}))$
EXP	$\sigma = exp(\frac{\gamma - Z}{1 - \sqrt{Z}}) \quad Z = \frac{\gamma HOL - \frac{1}{N_{RT}}\sum_{t=1}^{N_{RT}} \gamma HOL}{N_{RT}}$
MLWDF	$\sigma = -\frac{\log \delta}{\tau} T_{Max}$

A função exp(x) calcula o valor de e^x, a função log(x) retorna o logaritmo natural de x, HOL é o maior valor de atraso de pacotes de tempo real, Atraso desejado é o atraso máximo permitido para aquela classe, N_{RT} representa a quantidade de pacotes de tempo

real a serem enviados, T_{Max} representa o T máximo atingido por aquela antena, δ representa o tempo que o pacote passou na fila até o momento e τ representa o maior tempo que um pacote passou na fila.

Uma das maneiras apresentadas para a definição de parâmetros para o escalonamento é a utilização de *pricing models*. *Pricing model* é um modelo que segue diversas regras e atribuem um preço a um pacote. Esse preço pode ser entendido como o custo para a rede enviar o pacote ou o ganho que a rede tem se ele for enviado naquele instante. Diversos *pricing models* utilizam diferentes estratégias para criar o preço de um pacote. Em [Wallenius and Hamalainen 2002] os autores utilizam uma equação que se baseia em variáveis que descrevem o estado da rede como, por exemplo, vazão, atraso, *jitter* etc para elaborar o preço de cada um dos pacotes. A cada classe da rede, é atribuída uma prioridade. A ordem desta prioridade é: conversacional, *streaming*, interativas e de *background*. Além disso, também é criado um valor fixo para o custo de transmitir 1 bit daquele tipo de classe. O valor do preço dos bits é multiplicado pela constante que representa a prioridade da classe e esse é o preço para o envio daquele pacote. No entanto, o preço pode ser ajustado por um fator chamado de ”linearidade”pelos autores. Esse fator altera o preço do pacote se os recursos da rede estiverem escassos o preço do pacote sobe e ele pode não ser enviado no momento, porque o custo é alto para a rede.

Um *pricing model* tenta criar um equilíbrio entre a quantidade de recursos utilizados na rede e a qualidade das chamadas feitas. O trabalho apresentado em [Lai and Jiang 2014] utiliza um algoritmo genético que tenta encontrar a melhor solução de compromisso para o problema. Os autores utilizam duas equações para descrever o problema. Uma que descreve o lucro, ou seja, a quantidade de recursos que foi utilizada, e outra que descreve a satisfação dos usuários, ou seja, a qualidade de serviço da rede. O resultado do algoritmo genético é uma equação que descreve como cada classe de pacotes deve ter seu preço atribuído. Os resultados apresentados no artigo demonstram que as variáveis utilizadas nas equações são inversamente proporcionais, por isso é necessário encontrar um ponto de equilíbrio entre as duas. O algoritmo genético utilizado pelos autores consegue, a partir de um preço máximo para cada classe de serviço e uma satisfação mínima que deve ser atingida, criar um modelo que tenta balancear a satisfação e a utilização.

Em [Tarchi et al. 2006], os autores utilizam duas filas de prioridade para o envio dos pacotes. É definida uma prioridade para cada classe de serviço da rede e limites mínimos que ditam a porcentagem mínima de pacotes daquela classe devem ser enviados. A primeira fila de prioridades é a fila das classes de serviço. Enquanto o limiar de uma classe não for quebrado, a mais prioritária, com o limiar mais baixo é escolhida para ser enviada. Quando um limiar é quebrado aquela classe é escolhida para ser enviada enquanto o limiar não voltar para um nível aceitável. Após a escolha da classe, os autores escalonam os vários pacotes daquela classe. Eles utilizam técnicas clássicas como *round-robin* e *proportional fairness* para o envio dos pacotes.

Em [Ball et al. 2005], os autores utilizam uma fila simples de prioridade para escalonar os pacotes. A diferença apresentada no artigo é que os autores removem da fila pacotes que apresentam sinal de transmissão ruim. O autores fazem isso com a ideia de que pacotes que apresentam um nível ruim da qualidade de sinal tem grande chance de se perderem durante. Para evitar o envio de um pacote que será provavelmente perdido, os

autores o removem da lista por um tempo pré determinado e depois o colocam novamente. Esse processo é repetido algumas vezes enquanto o sinal for considerado ruim. Após um certo número de tentativas, o pacote é enviado de qualquer maneira.

Outra técnica utilizada é a apresentada em [Pizzi et al. 2009], na qual os autores estudam o que acontece se apenas um parâmetro da rede for utilizado para o escalonamento. Os pacotes são colocados em uma fila de prioridade e aqueles que possuem o maior *jitter* são enviados primeiro. O artigo demonstra que classes sensíveis ao *jitter* como a de *streaming* são extremamente beneficiadas por essa técnica. Contudo as classes que dependem de outros parâmetros da rede acabam sendo prejudicadas. Os autores em [Lai and Tang 2013] utilizam um modelo de de previsão de envio de pacotes para fazer o escalonamento. São utilizadas diversas variáveis de QoS nos cálculos do modelo como, por exemplo, atraso, *jitter* etc. O intuito do modelo é calcular se dado pacote que está na fila de envio chegará ou não a tempo, com atraso aceitável, ao seu destino. Uma vez que a previsão dos pacotes é feita, a fila de envio é reorganizada colocando os que atendem à condição na frente.

3. FuzSy: Escalonador de Pacotes baseado em QoS e Lógica *Fuzzy*

A ideia principal de um escalonador é despachar pacotes que chegam em elementos centralizadores de uma rede, tais como roteadores ou estações rádio base. Os escalonadores normalmente utilizam filas de pacotes e atribuem prioridades aos mesmos. Do mecanismo mais simples, FIFO, aos descritos anteriormente, a maioria segue regras estáticas de escolha de pacotes prioritários. Como em uma rede de celulares os picos de utilização dependem de horário, localização e dia da semana, os critérios estáticos podem significar uma distribuição injusta de recursos. Para tornar a distribuição mais justa, a prioridade de escalonamento deve ser alterada, dinamicamente, de acordo com a demanda do momento.

O FuzSy, ou *Fuzzy Scheduler*, se baseia no estado atual da rede e utiliza um mecanismo de lógica *fuzzy* para decidir qual classe de serviço é mais prioritária em dado instante. Para tomar essa decisão, o FuzSy utiliza metadados da rede, dados coletados durante a transmissão dos pacotes. Os metadados coletados são o atraso médio, que representa o tempo médio gasto para um pacote sair do dispositivo origem e chegar ao dispositivo destino, o *jitter* médio, que representa a diferença do atraso médio entre dois pacotes consecutivos da mesma aplicação, a vazão média, que representa a quantidade média de dados transmitidos em dado tempo, e a quantidade de pacotes a serem enviados. Cada uma das métricas citadas são coletadas separadamente para cada classe de serviço. As métricas coletadas são separadas por classe de serviço, porque cada uma destas classes apresenta um requisito diferente sobre essas métricas, ou seja, uma métrica tem maior impacto na qualidade daquele tipo de serviço do que em outro. Após a coleta dos metadados é necessário definir o que eles significam. Cada um dos metadados coletados possui três níveis, baixo, médio e alto. As Tabelas 2, 3, 4 e 5 possuem os valores utilizados para definição dos níveis.

Uma vez que todos os valores de entrada do sistema têm seus níveis definidos, são testadas as regras de inferência para decidir quais ações devem ser tomadas. Classicamente, num sistema de lógica *fuzzy*, cria-se todas as regras de inferência possíveis para o sistema, o que significaria criar 3^{16} regras neste sistema, o que é impraticável do ponto de vista de processamento, para isso foram criadas essas regras utilizando conjunções e

disjunções, para testar englobar todas as possíveis regras num conjunto menor. Na Tabela 6 estão algumas das regras utilizadas no sistema, por questões de espaço serão apresentadas apenas 5 das 27 regras utilizadas, mas todas seguem o mesmo padrão.

Tabela 2. Níveis de pertinência - Atraso médio

	Atraso médio (ms)		
	Baixo	Médio	Alto
Conversacional	0 - 75	25 - 100	50 - 150
Streaming	0 - 150	50 - 200	100 - 300
Interativa	1 - 150	50 - 200	101 - 300
Background	2 - 150	50 - 200	102 - 300

Tabela 3. Níveis de pertinência - *Jitter* médio

	Jitter Médio (ms)		
	Baixo	Médio	Alto
Conversacional	0 - 40	20 - 60	40 - 80
Streaming	0 - 20	10 - 30	20 - 40
Interativa	0 - 40	20 - 60	40 - 80
Background	0 - 40	20 - 60	40 - 80

Tabela 4. Níveis de pertinência - Vazão média

	Vazão Média (kbps)		
	Baixo	Médio	Alto
Conversacional	0 - 30	30 - 50	50 - 64
Streaming	0 - 350	150 - 665	500 - 1000
Interativa	0 - 150	100 - 250	200 - 350
Background	0 - 1500	1000 - 3000	2000 - 4500

Tabela 5. Níveis de pertinência - Quantidade de pacotes na fila

	Quantidade de pacotes (unidade)		
	Baixo	Médio	Alto
Conversacional	0 - 4	2 - 6	4 - 8
Streaming	0 - 4	2 - 6	4 - 8
Interativa	0 - 4	2 - 6	4 - 8
Background	0 - 4	2 - 6	4 - 8

Após o mecanismo de lógica *fuzzy* verificar todas as regras o processo de deffuzyficação é iniciado. Neste trabalho, foi utilizado o mecanismo chamado "maior de todos". Se mais de uma regra, para a mesma variável de saída, for atendida, a regra com o maior valor de pertinência é dada como válida e o valor é atribuído. Por exemplo, se uma regra disser que a variável deveria ser alta e outra regra disser que ela deveria ser muito alta o valor atribuído à variável é de muito alta. Durante as simulações notou-se que em ocasiões em que todas as variáveis de saída possuíam valores muito altos o escalonador não conseguia priorizar de forma correta o envio dos pacotes. Para isso, foi implementado um mecanismo estático de prioridades para ser executado apenas nesse caso. O mecanismo assumia que as classe seguiam a seguinte prioridade: conversacional,

streaming, interativa e *background* [3GPP 2013]. Uma vez que os níveis de prioridade das classes abaixassem, o mecanismo *fuzzy* entrava em ação novamente.

Tabela 6. Regras de inferência

	Regra
1	if (JitterStreaming is Alto and FilasStreaming is Alto) and (VazaoStreaming is any or DelayStreaming is any) then PrioridadeStreaming is MuitoAlto
2	if (JitterStreaming is Medio and FilasStreaming is Alto) and (VazaoStreaming is Baixo or DelayStreaming is Alto) then PrioridadeStreaming is Alto
3	if (JitterStreaming is Medio and FilasStreaming is Medio) and (VazaoStreaming is Baixo or DelayStreaming is Alto) then PrioridadeStreaming is Alto
4	if (JitterStreaming is Baixo and FilasStreaming is Baixo) and (VazaoStreaming is Baixo or DelayStreaming is Alto) then PrioridadeStreaming is Medio
5	if (JitterStreaming is Baixo and FilasStreaming is Baixo) and (VazaoStreaming is Alto or DelayStreaming is Baixo) then PrioridadeStreaming is Baixo

O processo descrito é executado para todos os pacotes que estão na fila de envio e, depois, a fila é reorganizada seguindo as prioridades definidas, uma prioridade maior é mais prioritária que uma menor. Sendo assim os pacotes que possuem nível de pertinência maior são os primeiros a saírem da fila.

4. Simulações e Resultados

Para avaliação do FuzSy, foram executadas simulações, utilizando o simulador LTE-Sim [Piro et al. 2011a]. O cenário de simulação foi composto por sete células de formato hexagonal. A quantidade inicial de usuários por célula foi 5 e essa quantidade foi sendo gradativamente aumentada até 25 usuários. Os usuários sempre possuíam acesso à rede e se movimentavam de acordo com um modelo de mobilidade aleatório a 3 km/h.

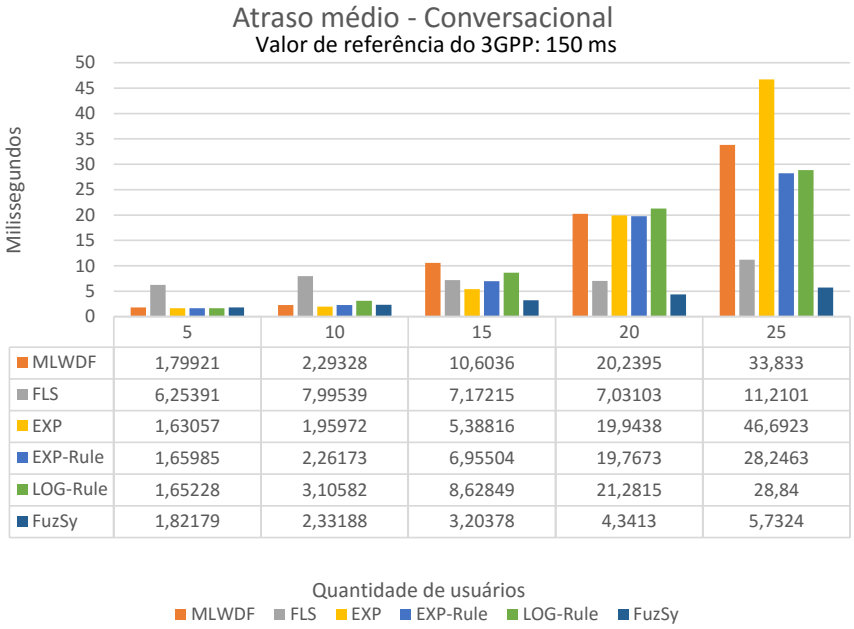
Tabela 7. Distribuição das aplicações

Classe	Distribuição
Conversacional	42%
Streaming	16%
Interativa	18.5%
Background	23.5%

Durante as simulações, foram criados quatro tipos de aplicações, uma para cada classe de serviço existente. Para representar a classe conversacional, foi simulada a transmissão de voz, pela rede, entre dois usuários. Para a classe *streaming*, foi utilizado a transmissão sob demanda em tempo real de um vídeo em 720p a partir de um servidor. Para a classe interativa, foi simulado o *download* de uma página da *web* com tamanho variando aleatoriamente entre 2 e 8 mb e para a classe *background*, foi simulado

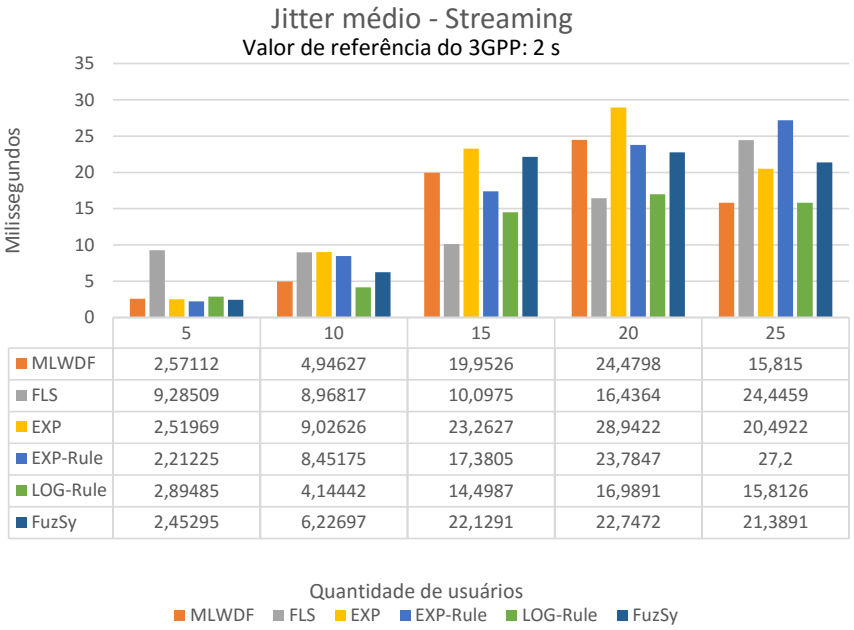
o download de um arquivo de um servidor com tamanho variando entre 10 e 50 mb. A distribuição das aplicações foi a mesma seguida em [Tostes Ribeiro et al. 2013], apresentada na Tabela 7, que representa uma distribuição mais próxima da realidade. Além dessa distribuição, também foi simulado um cenário com uma distribuição de 25% para cada tipo de aplicação e os resultados foram extremamente similares.

As métricas avaliadas foram o atraso médio dos pacotes, o *jitter* médio dos pacotes, a vazão média e a equidade média. Em todos os gráficos apresentados, os dados do escalonador *proportional fair* foram omitidos para melhor visualização, uma vez que ele sempre apresentava valores piores do que todos os outros escalonares avaliados, porque utiliza uma estratégia simples para fazer o escalonamento.



	5	10	15	20	25
MLWDF	1,79921	2,29328	10,6036	20,2395	33,833
FLS	6,25391	7,99539	7,17215	7,03103	11,2101
EXP	1,63057	1,95972	5,38816	19,9438	46,6923
EXP-Rule	1,65985	2,26173	6,95504	19,7673	28,2463
LOG-Rule	1,65228	3,10582	8,62849	21,2815	28,84
FuzSy	1,82179	2,33188	3,20378	4,3413	5,7324

Figura 1. Atraso médio - Conversacional



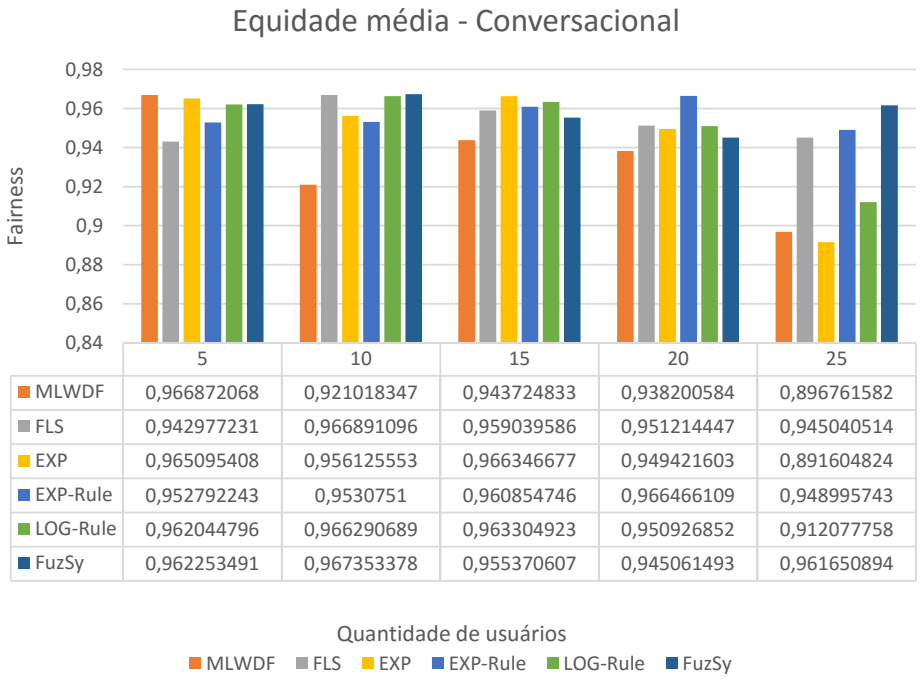
	5	10	15	20	25
MLWDF	2,57112	4,94627	19,9526	24,4798	15,815
FLS	9,28509	8,96817	10,0975	16,4364	24,4459
EXP	2,51969	9,02626	23,2627	28,9422	20,4922
EXP-Rule	2,21225	8,45175	17,3805	23,7847	27,2
LOG-Rule	2,89485	4,14442	14,4987	16,9891	15,8126
FuzSy	2,45295	6,22697	22,1291	22,7472	21,3891

Figura 2. *Jitter* médio - *Streaming*

No gráfico da Figura 1, pode-se observar o atraso médio para a classe conversacional. Pode-se perceber que o FuzSy apresentou resultados significativamente melhores que todos os outros, tendo valores máximos de aproximadamente 5 milissegundos para o cenário de 25 usuários que era o que apresentava maior utilização da rede.

O gráfico da Figura 2 mostra os valores de *jitter* para a classe *streaming*. O LOG-Rule foi o que apresentou melhores resultados neste quesito. O intuito do FuzSy não é conseguir os melhores valores em todos os parâmetros avaliados, deseja-se que ele consiga distribuir os recursos de forma mais justa, ou seja, sem prejudicar outras classes para priorizar uma. Devido a isso, ele não apresenta, necessariamente, o melhor ganho de desempenho, mas isso acontece para que seja possível garantir uma melhor distribuição dos recursos, o que o torna o mais justo de maneira geral.

No gráfico da Figura 3 estão os valores de equidade para a classe conversacional. Nele, é possível observar que para os cenários com até 20 usuários por célula todos os escalonadores conseguiram manter uma equidade acima de 90%. No entanto, no cenário com 25 usuários é possível perceber que o FuzSy apresentou os melhores resultados, seguido do EXP-Rule e, em seguida, pelo FLS. Isso demonstra que o FuzSy conseguiu lidar de uma maneira mais eficiente com a distribuição dos recursos para esta classe.



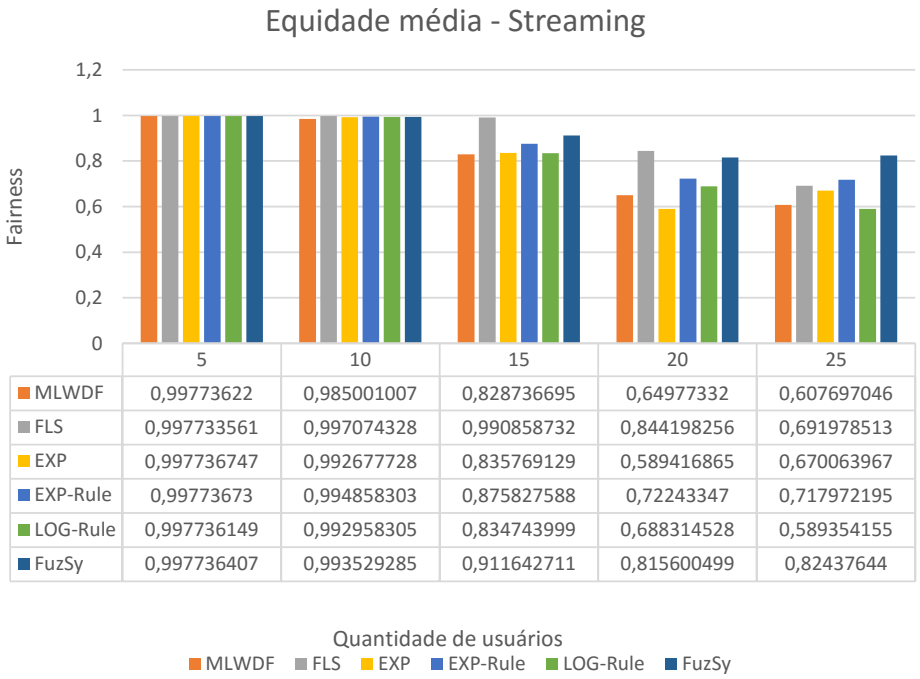
	5	10	15	20	25
MLWDF	0,966872068	0,921018347	0,943724833	0,938200584	0,896761582
FLS	0,942977231	0,966891096	0,959039586	0,951214447	0,945040514
EXP	0,965095408	0,956125553	0,966346677	0,949421603	0,891604824
EXP-Rule	0,952792243	0,9530751	0,960854746	0,966466109	0,948995743
LOG-Rule	0,962044796	0,966290689	0,963304923	0,950926852	0,912077758
FuzSy	0,962253491	0,967353378	0,955370607	0,945061493	0,961650894

Figura 3. Equidade média - Conversacional

Os valores de equidade média para a classe *streaming* estão no gráfico da Figura 4. É possível observar que, nos cenários até 10 usuários todos os escalonares ficaram muito próximos a 100% de equidade. A partir do cenário com 15 usuários esses valores começam a diminuir, sendo que para o cenário com 25 usuários, o mais denso, o FuzSy conseguiu uma equidade maior do que todos os outros escalonadores.

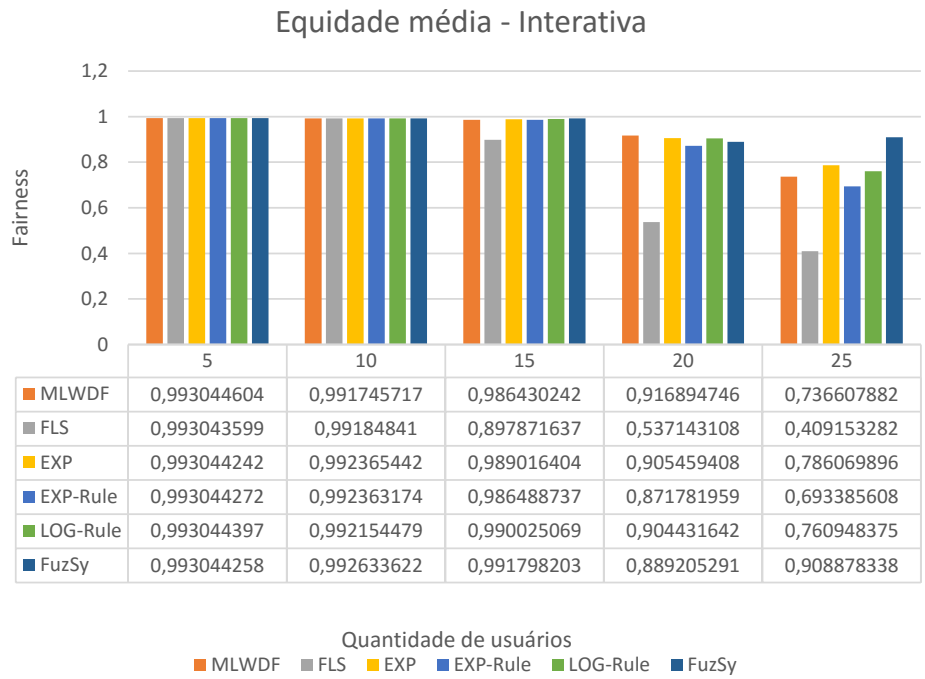
É possível observar os valores de equidade média para a classe interativa no gráfico da Figura 5. Os valores de equidade para o escalonador FLS caem significativamente para os cenários com um número maior de usuários por células. Isso acontece

devido à prioridade que este escalonador dá às classes conversacional e *streaming*. Com isso a classe interativa acaba ficando prejudicada. Diferentemente o FuzSy apresentou os maiores valores de equidade para os cenários com mais usuários por células, não prejudicando nenhuma classe, tentando manter parâmetros de QoS relevantes em níveis adequados para todas as classes.



	5	10	15	20	25
MLWDF	0,99773622	0,985001007	0,828736695	0,64977332	0,607697046
FLS	0,997733561	0,997074328	0,990858732	0,844198256	0,691978513
EXP	0,997736747	0,992677728	0,835769129	0,589416865	0,670063967
EXP-Rule	0,99773673	0,994858303	0,875827588	0,72243347	0,717972195
LOG-Rule	0,997736149	0,992958305	0,834743999	0,688314528	0,589354155
FuzSy	0,997736407	0,993529285	0,911642711	0,815600499	0,82437644

Figura 4. Equidade média - *Streaming*

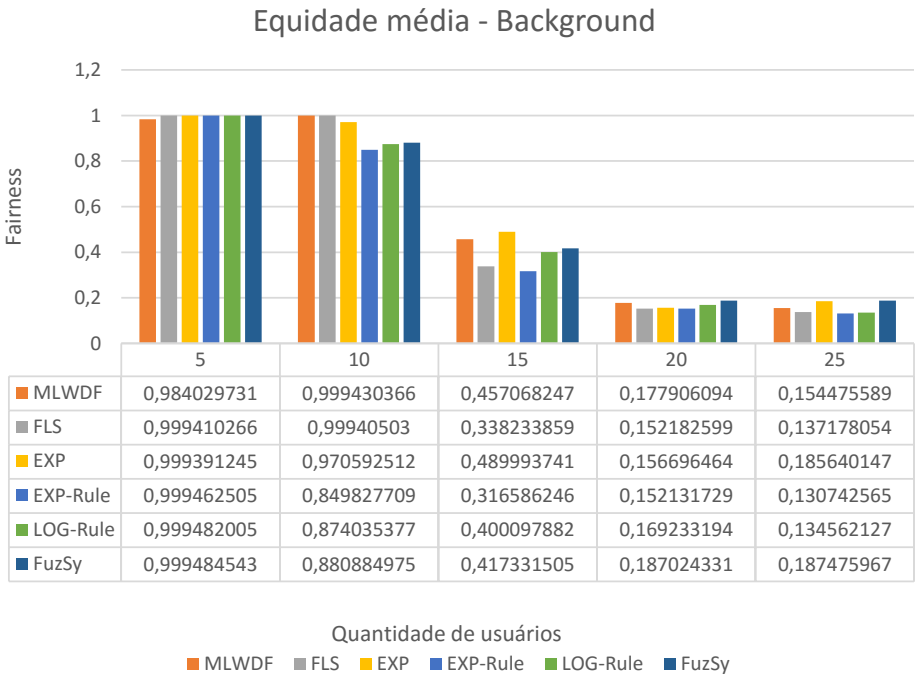


	5	10	15	20	25
MLWDF	0,993044604	0,991745717	0,986430242	0,916894746	0,736607882
FLS	0,993043599	0,99184841	0,897871637	0,537143108	0,409153282
EXP	0,993044242	0,992365442	0,989016404	0,905459408	0,786069896
EXP-Rule	0,993044272	0,992363174	0,986488737	0,871781959	0,693385608
LOG-Rule	0,993044397	0,992154479	0,990025069	0,904431642	0,760948375
FuzSy	0,993044258	0,992633622	0,991798203	0,889205291	0,908878338

Figura 5. Equidade média - Interativa

O gráfico da Figura 6 apresenta os valores de equidade média para a classe *background*. É possível observar que, à medida que a quantidade de usuários aumenta, a

equidade em todos os escalonadores diminui vertiginosamente. Isso acontece devido à disputa por recursos ser maior nestes cenários, com uma quantidade maior de usuários utilizando a rede os recursos que seriam alocados para esta classe são alocados para as aplicações das outras classes.



	5	10	15	20	25
MLWDF	0,984029731	0,999430366	0,457068247	0,177906094	0,154475589
FLS	0,999410266	0,99940503	0,338233859	0,152182599	0,137178054
EXP	0,999391245	0,970592512	0,489993741	0,156696464	0,185640147
EXP-Rule	0,999462505	0,849827709	0,316586246	0,152131729	0,130742565
LOG-Rule	0,999482005	0,874035377	0,400097882	0,169233194	0,134562127
FuzSy	0,999484543	0,880884975	0,417331505	0,187024331	0,187475967

Figura 6. Equidade média - *Background*

Na Tabela 8 estão as posições alcançadas pelos escalonadores quando comparados em relação aos parâmetros mais sensíveis de cada classe e também em relação à equidade, as letras após o nome do parâmetro representam a inicial de cada uma das classes. É possível perceber que, para a maioria dos parâmetros comparados, o FuzSy foi o escalonador com o melhor desempenho. Isso acontece graças a habilidade do FuzSy de se adaptar melhor às diversas situações que podem ocorrer. Ele não foi o escalonador com melhores resultados de *jitter* para a classe *streaming*, mas conseguiu garantir uma equidade maior do que todos os outros escalonadores.

Tabela 8. Ranking dos escalonadores para cenário com 25 usuários por célula

	1º	2º	3º	4º	5º	6º
Vazão I	FuzSy	EXP	LOG-Rule	MLWDF	EXP-Rule	FLS
Atraso C	FuzSy	FLS	EXP-Rule	LOG-Rule	MLWDF	EXP
Jitter S	LOG-Rule	MLWDF	EXP	FuzSy	FLS	EXP-Rule
Equidade C	FuzSy	EXP-Rule	FLS	LOG-Rule	MLWDF	EXP
Equidade S	FuzSy	EXP-Rule	FLS	EXP	MLWDF	LOG-Rule
Equidade I	FuzSy	EXP	LOG-Rule	MLWDF	EXP-Rule	FLS
Equidade B	FuzSy	EXP	MLWDF	FLS	LOG-Rule	EXP-Rule

5. Conclusão

O FuzSy se baseia no estado atual da rede e busca garantir a equidade entre os requisitos de QoS das classes de serviço da rede, através de controle dinâmico implementado por

meio de um mecanismo de lógica *fuzzy*. Utilizando metadados, colhidos durante o envio dos pacotes, o mecanismo de lógica *fuzzy* é executado para definir a prioridade dos pacotes que devem ser enviados. Tendo como entrada, para este mecanismo, parâmetros definidos através de simulações exaustivas e recomendações do órgão responsável pela padronização de redes 4G, ele consegue saber quais classes de serviço estão em maior risco de não atingirem seus limites mínimos de QoS. As regras de inferência criadas tentam fazer com que os parâmetros mais importantes para cada classe de serviço sejam levados em consideração, fazendo com que estes parâmetros sejam utilizados para melhorar a equidade entre as classes. A avaliação do escalonador *fuzzy* foi feita através de simulações realizadas no LTE-SIM. Ao ser comparado com seis outros métodos da literatura, o FuzSy apresentou resultados para todas as métricas dentro dos limites sugeridos pelo 3GPP e apresentou também melhorias na equidade das classes.

O diferencial do FuzSy, em relação aos outros escalonadores simulados, é a utilização de uma prioridade dinâmica para as classes. Os escalonadores avaliados definem quais classes de tempo real têm maior prioridade sobre as outras classes, o que pode causar má distribuição dos recursos. Foi demonstrado, através dos resultados de simulação, que a utilização de prioridades estáticas para as classes *streaming* e conversacional levaram o escalonador FLS a prejudicar a classe interativa, o que não acontece com o FuzSy. Sendo dinâmico, ele permite a utilização mais justa da rede, sem prejudicar o desempenho das mais prioritárias. Isso significa que há uma utilização mais efetiva e justa dos recursos da rede.

A utilização de um mecanismo de lógica *fuzzy* permite que os limites estabelecidos para cada classe sejam facilmente modificados e as regras de inferência também. O escalonador pode ser adaptado a qualquer situação específica que de uma operadora, como, por exemplo, eventos de larga escala, onde predomina um tipo ou dois tipos de aplicações ou classes de serviço. O FuzSy pode ser considerado mais justo do que os outros escalonadores comparados, neste trabalho, porque ele privilegia as classes mais prioritárias, mas além disso, redistribui os recursos de maneira a não permitir que o desempenho das outras classes caia.

Como trabalho futuro, sugere-se a imlementação de um algoritmo genético, que leve em consideração o estado final das simulações como função objetivo. Ele pode ser utilizado para melhorar a escolha dos parâmetros que definem os níveis de pertinência de cada uma das variáveis. A utilização de modelos de mobilidade que reflitam mais a movimentação de usuários comuns como, por exemplo, usuários em automóveis e transporte público e simulações com uma maior aleatoriedade na duração de cada tipo de chamada e a utilização de outros mecanismos de garantia de qualidade de serviço em conjunto com o escalonador como controles de admissão de chamadas, também são trabalhos futuros.

Agradecimentos

Agradecemos à CAPES e à Microsoft Research, sem as quais este trabalho não poderia ter sido executado.

Referências

3GPP (2013). Lte release 10. http://www.3gpp.org/technologies/keywords-acronyms/97-lte-advanced. Acessado em 10.04.15.

Andrews, M., Kumaran, K., Ramanan, K., Stolyar, A., Whiting, P., and Vijayakumar, R. (2001). Providing quality of service over a shared wireless link. *IEEE Communications Magazine*, 39(2):150–154.

Ball, C., Treml, F., Gaube, X., and Klein, A. (2005). Performance analysis of temporary removal scheduling applied to mobile wimax scenarios in tight frequency reuse. In *Personal, Indoor and Mobile Radio Communications, 2005. PIMRC 2005. IEEE 16th International Symposium on*, volume 2, pages 888–894 Vol. 2.

de Souza, F. R. (2016). Fuzsy: um escalonador de pacotes baseado em qualidade de serviço e lógica fuzzy. Master's thesis, Pontifícia Universidade Católica de Minas Gerais.

IBGE (2016). Projeções e estimativas da população do brasil e das unidades da federação. Disponível em: <http://www.ibge.gov.br/apps/populacao/projecao/notatecnica.html>.

Jalali, A., Padovani, R., and Pankaj, R. (2000). ata throughput of cdma-hdr a high efficiency-high data rate personal communication wireless system. In *Vehicular Technology Conference Proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st*, volume 3, pages 1854–1858 vol.3.

Lai, W. K. and Tang, C.-L. (2013). Qos-aware downlink packet scheduling for {LTE} networks. *Computer Networks*, 57(7):1689 – 1698.

Lai, Y.-L. and Jiang, J.-R. (2014). Pricing resources in lte networks through multiobjective optimization. volume 2014, page 8.

Piro, G., Grieco, L., Boggia, G., Capozzi, F., and Camarda, P. (2011a). Simulating lte cellular systems: An open-source framework. *Vehicular Technology, IEEE Transactions on*, 60(2):498–513.

Piro, G., Grieco, L. A., Boggia, G., Fortuna, R., and Camarda, P. (2011b). Two-level downlink scheduling for real-time multimedia services in lte networks. *IEEE Transactions on Multimedia*, 13(5):1052–1065.

Pizzi, S., Molinaro, A., and Iera, A. (2009). On the performance of "compensation-based"and "greedy"scheduling policies in ieee 802.16 networks. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1–6.

Rhee, J.-H., Holtzman, J. M., and Kim, D. K. (2004). Performance analysis of the adaptive exp/pf channel scheduler in an amc/tdm system. *IEEE Communications Letters*, 8(8):497–499.

Sadiq, B., Baek, S. J., and de Veciana, G. (2011). Delay-optimal opportunistic scheduling and approximations: The log rule. *Networking, IEEE/ACM Transactions on*, 19(2):405–418.

Shakkottai, S. and Stolyar, A. L. (2002). Scheduling for multiple flows sharing a time-varying channel: The exponential rule. *Translations of the American Mathematical Society-Series 2*, 207:185–202.

Tarchi, D., Fantacci, R., and Bardazzi, M. (2006). Quality of service management in ieee 802.16 wireless metropolitan area networks. In *Communications, 2006. ICC '06. IEEE International Conference on*, volume 4, pages 1789–1794.

Tostes Ribeiro, A. I. J., Zarate, L. E., and Duarte Figueiredo, F. d. L. P. (2013). Dynamic fuzzy cellular admission control. In *5th IEEE Latin-American Conference on Communications (IEEE LATINCOM 2013)*.

Wallenius, E. and Hamalainen, T. (2002). Pricing model for 3g/4g networks. In *Personal, Indoor and Mobile Radio Communications, 2002. The 13th IEEE International Symposium on*, volume 1, pages 187–191 vol.1.

Otimização de Recursos Energéticos em Sistemas SC-FDMA com Garantias de QoS e Satisfação

Iran M. B. Júnior[1], **F. Rafael M. Lima**[12], **Tarcisio F. Maciel**[2], **F. Rodrigo P. Cavalcanti**[2]

[1]Engenharia da Computação, Universidade Federal do Ceará (UFC),
Sobral, Ceará, Brasil

[2]Grupo de Pesquisa em Telecomunicações Sem Fio (GTEL),
Universidade Federal do Ceará, Fortaleza, Ceará, Brasil

iranjunior51@hotmail.com, {rafaelm,maciel,rodrigo}@gtel.ufc.br

Abstract. *This paper proposes an optimal solution to the frequency and power resource allocation employing Single Carrier - Frequency Division Multiple Access (SC-FDMA) such as in Long Term Evolution (LTE). The main objective is to minimize the total transmitted power in a cellular and multiservice system with constraints on the percentage of users that should be satisfied per service. Through algebraic manipulations we were able to convert an integer nonlinear optimization problem in an Integer Linear Problem (ILP) so as to obtain the optimal solution. Results of computer simulations show the importance of adaptive power allocation in the context of wireless cellular networks.*

Resumo. *Este trabalho propõe uma solução ótima para alocação de recursos em termos de frequência e potência de transmissão em sistemas SC-FDMA (do inglês, Single Carrier - Frequency Division Multiple Access) tal qual empregado no sistema LTE (do inglês, Long Term Evolution). Estudamos o problema de minimização da potência total transmitida de um sistema celular multisserviços sujeito a restrições que assegurem um percentual mínimo de usuários satisfeitos. Através de manipulações algébricas, convertemos um problema inteiro não-linear em um problema inteiro e linear a fim de obter a solução ótima. Os resultados obtidos através de simulações computacionais mostram a importância da alocação adaptativa de potência no contexto de redes celulares sem fio.*

1. Introdução

O uso de dispositivos sofisticados como *smartphones*, *tablets* e *notebooks* têm proporcionado uma rápida popularidade das novas aplicações móveis e serviços, provocando também um aumento na demanda por altas taxas de dados em rede móveis. Além disso, em um futuro próximo, espera-se que todos os objetos do dia a dia, tais como geladeiras, televisores e outras máquinas, possam se conectar a *internet*. Assim, a busca por um menor consumo de energia juntamente com o aumento de tráfego e a necessidade de prover qualidade de serviço sustentável, surgem como um cenário desafiador para as operadoras dos sistemas e para indústria de telecomunicações. O desenvolvimento de novas estratégias que possam atender a essas novas demandas de tráfego, assim como proporcionar uma melhor eficiência energética, estão de acordo com a evolução das várias gerações das comunicações móveis.

O sistema LTE e LTE-*Advanced* são sistemas capazes de atender a crescente demanda de tráfego em redes móveis. A fim de cumprir essa tarefa com êxito, estes sistemas empregam no enlace direto o esquema de múltiplo acesso ou OFDMA (do inglês, *Orthogonal Frequency Division Multiple Access*). Este esquema de múltiplo acesso proporciona muitas vantagens tais como alta eficiência espectral, granularidade na alocação de recursos e proteção contra os efeitos danosos causados pelo canal de multipercurso tais como a interferência inter-simbólica. No entanto, o esquema OFDMA não é utilizado no enlace reverso por apresentar uma grande variabilidade na relação entre a potência de pico e potência média ou PAPR (do inglês, *Peak-to-Average Power Ratio*), o que torna necessário o uso de amplificadores de potência altamente lineares nos transmissores (terminais móveis). Isso traria problemas quanto ao custo e tamanho dos terminais móveis.

Diante disso, o esquema de múltiplo acesso por divisão de frequência com portadora única ou SC-FDMA foi escolhido como esquema de múltiplo acesso do enlace reverso ou *uplink* dos sistemas LTE/LTE-A. Comparado ao OFDMA, o SC-FDMA possui uma baixa PAPR [Myung et al. 2006]. Porém, sinais SC-FDMA podem apresentar valores consideráveis de interferência intersimbólica ou ISI (do inglês, *Intersymbol Interference*) nas ERBs (Estações de Rádio Base). Desta forma, deve-se empregar um equalizador adaptativo no domínio da frequência na ERB a fim de minimizar a interferência intersimbólica.

Outra diferença crucial entre os esquemas SC-FDMA e OFDMA é a adjacência de recursos na frequência ao alocar recursos aos terminais móveis necessária para garantir baixos valores de PAPR. Diferentemente dos sistemas OFDMA em que diferentes blocos de recursos localizados em diferentes pontos da banda de frequência disponível podem ser alocados livremente para transmissão, em SC-FDMA, os blocos de recursos na frequência devem ser alocados de forma contígua ou adjacente na frequência para cada terminal móvel, o que dificulta a tarefa de alocação de recursos de rádio nestas redes.

Neste artigo, estudamos sistemas SC-FDMA tal qual empregado no enlace reverso dos sistemas LTE/LTE-A sob a ótica de alocação de recursos de rádio com objetivo de minimizar a potência transmitida no enlace reverso e garantir satisfação dos usuários do sistema em termos de QoS (do inglês, *Quality of Service*). O restante deste artigo está organizado da seguinte forma. Na seção 2, revisamos brevemente os trabalhos mais relevantes relacionados ao nosso artigo e apresentamos as principais contribuições deste trabalho. Nas seções 3 e 4, apresentamos os principais pressupostos referentes à modelagem do sistema e também o problema estudado na forma de otimização, respectivamente. Na seção 5 apresentamos métodos para obter a solução ótima do problema estudado. Um estudo de caso é discutido na seção 6, onde demonstramos os ganhos resultantes da gerência de recursos de rádio em sistemas LTE/LTE-A. Finalmente, as principais conclusões e perspectivas estão resumidas na seção 7.

2. Revisão Bibliográfica e Contribuições

Em esquemas OFDMA, os problemas de RRA (do inglês, *Radio Resource Allocation*) têm sido estudados na forma de otimização com diferentes restrições e objetivos. [Capozzi et al. 2013] é uma boa referência para um leitor que deseja se aprofundar um pouco mais sobre este tópico. Diferentemente dos sistemas OFDMA, os estudos de RRA para sistemas SC-FDMA são bem mais recentes.

Na literatura, a *maximização da taxa total de dados* é um dos problemas mais comuns em RRA. Este problema consiste em encontrar a alocação de recurso que maximiza a taxa de dados total transmitida. Para sistemas OFDMA, tal qual o enlace direto dos sistemas LTE/LTE-A, este problema pode ser resolvido de forma ótima utilizando um simples algoritmo de complexidade computacional polinomial proposto em [Jang and Lee 2003]. No entanto, como consequência da restrição de adjacência, a obtenção da solução ótima deste problema apresenta-se como uma tarefa bem mais complexa em sistemas SC-FDMA. Em [Lim et al. 2006] foi considerado o problema de *maximização da taxa total de dados* no cenário SC-FDMA no enlace reverso. Porém, a restrição de adjacência necessária para assegurar uma baixa PAPR foi ignorada pelo trabalho. Em [Nwamadi et al. 2008], os autores consideraram a restrição de adjacência, todavia, assumiu-se que cada usuário exigia uma certa quantidade de recursos de frequência. Na prática, usuários possuem requisitos de QoS distintos que podem ser diferentes taxas de dados requisitadas, por exemplo. Além disso, as qualidades de canal experimentada pelos enlaces de dados dos usuários em geral podem ser consideradas independentes do ponto de vista estatístico. Como o número de recursos em frequência alocados a cada usuário depende tanto dos requisitos de QoS como da qualidade do canal, podemos concluir que a modelagem do artigo [Nwamadi et al. 2008] carece de apelo prático.

Recentemente, um dos problemas que vem ganhando espaço em RRA consiste na maximização da eficiência energética em sistemas sem fio. Este tipo de problema é motivado pela crescente demanda por maior eficiência no uso dos recursos energéticos em rede móveis. De uma forma geral, estes problemas tem por objetivo minimizar a potência total transmitida ou minimizar o consumo de energia por bit transmitido. Um dos primeiros trabalhos nessa direção consistiu na minimização da potência total transmitida sujeita a restrições de QoS para sistemas OFDMA presente no trabalho [Wong et al. 1999]. Já em [Triantafyllopoulou et al. 2015], os autores consideraram os efeitos de QoS sobre a eficiência energética com o objetivo minimizar o consumo de energia por bit transmitido no sistema SC-FDMA. Embora os aspectos de QoS tenha sido considerados no trabalho, os autores não utilizaram alocação de potência adaptativa e ignoraram o cenário multisserviços.

Atualmente, uma gama de serviços são oferecidos para os diversos dispositivos móveis. Os trabalhos [Ruby and Leung 2011] e [Delgado and Jaumard 2010] consideram um cenário multisserviço em que é assumido que os terminais móveis podem estar utilizando diferentes aplicações móveis e portanto, terem requisitos de QoS diferentes. Por exemplo, um terminal móvel pode estar fazendo a transmissão de um arquivo pessoal para um servidor na nuvem, enquanto que outro terminal participa de uma conversação por vídeo-telefonia. O trabalho [Delgado and Jaumard 2010] também aborda um problema de RRA em cenários multisserviço maximizando uma função de utilidade restrita aos requisitos de QoS de diferentes serviços ou classes de tráfego. Os autores propõem dois algoritmos heurísticos para resolver o problema, mas não fornece uma solução ótima ou limitante superior em desempenho.

Embora as questões de QoS para o cenário multisserviço para sistemas SC-FDMA tenham sido abordadas nos trabalhos previamente citados, as garantias de satisfação mínima por serviço foram ignoradas pelos mesmo. Definimos satisfação por serviço como o percentual (ou número) de usuários que utilizam um certo tipo de serviço que

devem ser efetivamente satisfeitos. A operadora móvel pode definir de forma flexível esse percentual de forma a realizar controle de carga e congestionamento. Por exemplo, a operadora móvel pode determinar que o número de usuários satisfeitos em vídeo-telefonia seja maior que o número de usuários satisfeitos que fazem *upload* de arquivos na nuvem, por entender que o primeiro serviço é mais prioritário. As garantias de satisfação mínima foram propostas como métricas de nível sistêmico em [Furuskär 2003] e usadas em muitas outros trabalhos, como [Lima et al. 2010] e [Maurício et al. 2014].

Em [Lima et al. 2016], dois problemas distintos de RRA em sistemas SC-FDMA foram considerados: *maximização da taxa de dados* ou problema URM (do inglês, *Unconstrained Rate Maximization*) e *maximização da taxa de dados sujeito aos requisitos de satisfação mínima* em um cenário multisserviço ou problema CRM (do inglês, *Constrained Rate Maximization*). Em ambos os problemas a restrição de adjacência foi modelada. No entanto, os autores consideraram que os terminais móveis transmitiam com toda potência disponível, não considerando portanto, alocação adaptativa de potência. Essa consideração permite que o sistema alcance altos valores de taxas de dados, no entanto, se torna ineficiente quanto ao consumo de energia. Além disso, o mapeamento contínuo entre taxa de dados e SNR (do inglês, *Signal-to-Noise Ratio*) foi considerado em ambos os problemas. Esta última hipótese traz benefícios teóricos por simplificar a estrutura do problema de RRA, contudo, perde apelo prático uma vez que a adaptação entre taxa de dados e SNR é regida por esquemas finitos e discretos de modulação e codificação ou MCS (do inglês, *Modulation and Coding Scheme*).

Diante do que foi exposto nesta seção, as principais contribuições deste artigo são:

- Formulação de um novo de problema de otimização dos recursos energéticos considerando as restrições de adjacência e de QoS presente em [Lima et al. 2016].
- Proposta de solução ótima para o novo problema após uma reformulação que transformou um problema combinatorial não linear em ILP (do inglês, *Integer Linear Problem*).
- Análise de desempenho da solução para o novo problema proposto considerando como cenário de estudo de caso o enlace reverso do sistema LTE. Em particular, enfatizamos o impacto e potenciais ganhos em termos energéticos no problema de RRA apresentado em [Lima et al. 2016].

3. Modelagem do Sistema

Assumimos que a alocação de recursos deve ser realizada em um setor da célula de um sistema celular com uma ERB que serve os terminais conectados. Ambos terminais móveis e ERB empregam transceptores de antena única. Para este trabalho iremos considerar um problema no enlace reverso da rede ou *uplink*. Utilizaremos a combinação SC-FDMA e TDMA para controlar a interferência intracelular entre o usuários do mesmo setor. Definimos um bloco de recurso ou RB (do inglês, *Resource Block*), como o mínimo recurso alocável a um determinado usuário, que é composto por um grupo de uma ou mais subportadoras OFDM adjacentes e um número de símbolos OFDM consecutivos, que representam um intervalo de tempo de transmissão ou TTI (do inglês, *Transmission Time Interval*). Supomos que a interferência intercelular, ou seja a interferência causada pelo reuso da mesma banda de frequência em outros setores, é modelada como uma variável Gaussiana e que a mesma faz parte do ruído térmico na expressão da SNR. Ressaltamos

que essa suposição se torna cada vez mais válida à medida que a carga do setor e o número de células no sistema aumentam [Seol et al. 2010].

Nosso foco neste artigo é sobre um problema de atribuição de recurso instantâneo que consiste em determinar em um determinado TTI, quais recursos devem ser atribuídos aos terminais conectados a fim de cumprir objetivos e restrições específicas (definidos mais adiante na seção 4). Observe que a solução sequencial do problema de atribuição de recursos ao longo de vários TTIs é equivalente a um problema de escalonamento de pacotes. Portanto, consideramos que J usuários são candidatos a receber recursos e que existe um total de N RBs disponíveis. Além disso, $\mathcal{J}$ e $\mathcal{N}$ são os conjuntos de usuários ativos e RBs, respectivamente.

Assumimos que o operador do sistema fornece diferentes serviços aos usuários ativos, tais como *web browsing*, *download/upload* de arquivos ou VoIP (do inglês, *Voice over Internet Protocol*). Assumimos que existe um total de S serviços no sistema e $\mathcal{S}$ representa o conjunto de todos os serviços disponibilizados pela operadora móvel. Consideramos que existem J_s usuários utilizando o serviço $s \in \mathcal{S}$ e $\mathcal{J}_s$ consiste no conjunto de usuários que utilizam o serviço s. Assumimos também que, $\cup_{s\in\mathcal{S}}\ \mathcal{J}_s = \mathcal{J}$ e $\sum_{s\in\mathcal{S}}\ J_s = J$.

Consideramos que os coeficientes do canal rádio móvel permanecem aproximadamente constantes durante um TTI e que no TTI atual, o usuário j possui um requisito de taxa igual a t_j . Uma observação importante é que, requisitos de taxa de longo ou médio prazo podem ser mapeados em requisitos instantâneos conforme mostrado em [Lima et al. 2010]. O parâmetro k_s representa o número mínimo de usuários que devem ser satisfeitos com seu QoS para cada serviço.

O sistema SC-FDMA impõe duas restrições sobre assinalamento de recursos conforme explicado antes. A primeira, denominada por restrição de exclusividade, garante que um mesmo RB não pode ser reutilizado dentro de uma célula. Note que esta restrição também está presente em sistemas OFDMA. A segunda consiste na restrição de adjacência, a qual determina que os RBs atribuídos a um dado usuário devem ser adjacentes uns aos outros no domínio da frequência. Esta restrição é própria do SC-FDMA e necessária para obter benefícios em termos de PAPR.

O número de possíveis assinalamentos do sistema é limitado em virtude da restrição de adjacência. De acordo com o trabalho [Wong et al. 2009], o número de padrões de assinalamentos, P, que podem ser formados com N RBs é dado por

$$P = N^2/2 + N/2 + 1. \tag{1}$$

Como exemplo, considerando $N = 4$, temos $P = 11$ e os possíveis grupos de RBs ou padrões de assinalamentos são $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{1,2\}$, $\{2,3\}$, $\{3,4\}$, $\{1,2,3\}$, $\{2,3,4\}$, $\{1,2,3,4\}$ e $\{\emptyset\}$. Assumimos $\mathcal{P}$ como o conjunto com todos os índices de todos os padrões de assinalamentos de recursos. De acordo com isto, ao invés de modelarmos a atribuição de um RB específico ao usuário, consideramos neste trabalho o assinalamento de um padrão de recursos a um terminal. Podemos, então, modelar a restrição de adjacência por uma matriz binária $\mathbf{A}$ com elementos $a_{n,p}$, com $n \in \mathcal{N}$ e $p \in \mathcal{P}$, que assume 1 se o RB n pertence ao padrão de assinalamento p e 0 caso contrário.

Tabela 1. Mapeamento entre taxa de dados e SNR

Região de SNR	Taxa de dados transmitida
$\gamma^1 \leq \gamma < \gamma^2$	r_1
$\gamma^2 \leq \gamma < \gamma^3$	r_2
$\vdots$	$\vdots$
$\gamma^{M-1} \leq \gamma < \gamma^M$	r_{M-1}
$\gamma^M \leq \gamma$	r_M

É comum na literatura considerar o mapeamento contínuo entre taxa de dados e SNR. Em geral, essas funções apresentam a propriedade de convexidade. Matematicamente, esta abordagem é vantajosa, pois existem alguns teoremas baseados na continuidade e convexidade das funções objetivos e das restrições que nos possibilitam obter uma solução analítica para estes problemas. Contudo, na prática, a função responsável pelo mapeamento entre taxa de dados e SNR é discreta e pode ser obtida através de simulações de nível de enlace e camanda física. Assim, na busca de um cenário mais realista, o mapeamento discreto será considerado neste trabalho.

Consideramos M níveis de MCSs contidos no conjunto $\mathcal{M} = \{1, \ldots, M\}$. Para que um determinado usuário atinja o nível de MCS m e portanto transmita com taxa de dados r_m, é necessário que a SNR experimentada esteja no intervalo de SNR [γ^m, γ^{m+1}], em que $\gamma^m < \gamma^{m+1}$, conforme a Tabela 1. Assumimos que a máxima potência disponível no terminal j para transmissão é P_{T_j}.

A SNR experimentada pelo enlace entre a ERB e o usuário j quando transmite sobre a z-ésima subportadora do RB n, $\gamma_{j,z,n}$, é dada por

$$\gamma_{j,z,n} = ((p_{j,p,m}/(c \cdot N_p)) \cdot \alpha_j \cdot ||h_{j,z,n}||^2)/\sigma^2 = p_{j,p,m} \cdot \overline{g}_{j,z,n}, \tag{2}$$

em que $p_{j,p,m}$ é a potência necessária para o usuário j utilizando o padrão de assinalamento p atingir o nível de MCS m, c é o numero de subportadoras em um RB, N_p o número de RBs do padrão de assinalamento p, α_j representa o efeito conjunto da perda de percurso e sombreamento do enlace entre o usuário j e a ERB, σ^2 é a potência do ruído no receptor na largura de banda de uma subportadora, $h_{j,z,n}$ é a resposta complexa em frequência do canal entre a ERB e o usuário j sobre a z-ésima subportadora do RB n e, finalmente, $\overline{g}_{j,z,n}$ é o ganho de canal total do enlace entre a ERB e o usuário j na z-ésima subportadora do RB n normalizado pela potência do ruído térmico. Definimos $|| \cdot ||$ como o operador que retorna o valor absoluto do seu argumento.

Como visto anteriormente, o sistema SC-FDMA necessita de equalizador no domínio da frequência na ERB para combater a ISI. Para este trabalho, utilizaremos o equalizador MMSE (do inglês, *Minimum Mean Square Error*), e de acordo com [Shi et al. 2004], a SNR efetiva experimentada pelo receptor quando os dados são transmitidos pelo usuário j utilizando o padrão de assinalamento p, $\gamma_{j,p}^{MMSE}$, é dada por

$$\gamma_{j,p}^{MMSE} = \left(\left(\frac{1}{c \cdot |\mathcal{N}_p|} \sum_{n \in \mathcal{N}_p} \sum_{z=1}^{c} \frac{\gamma_{j,z,n}}{\gamma_{j,z,n} + 1} \right)^{-1} - 1 \right)^{-1}, \tag{3}$$

em que $\mathcal{N}_p$ é o conjunto de RBs que compõem o padrão de assinalamento p e $|\cdot|$ denota a cardinalidade de um conjunto.

Assuma $f(\cdot)$ correspondente a função responsável pelo mapeamento discreto apresentado na Tabela 1. A taxa transmitida quando o usuário j utilizando o padrão de assinalamento p atinge o nível de MCS m, é dada por $r_{j,p,m} = f(\gamma_{j,p}^{MMSE})$.

Substituindo a equação (2) em (3), e fazendo $\gamma_{j,p}^{MMSE} = \gamma^m$, podemos encontrar os valores para $p_{j,p,m}$ através da solução numérica da equação a seguir:

$$\gamma^m = \left(\left(\frac{1}{c \cdot |\mathcal{N}_p|} \sum_{n \in \mathcal{N}_p} \sum_{z=1}^{c} \frac{p_{j,p,m} \cdot \overline{g}_{j,z,n}}{p_{j,p,m} \cdot \overline{g}_{j,z,n} + 1} \right)^{-1} - 1 \right)^{-1}. \tag{4}$$

4. Formulação do Problema

O problema formulado nesse trabalho consiste em minimizar a potência total transmitida no enlace reverso de um sistema empregando SC-FDMA utilizando as restrições de satisfação mínima de usuários por serviço em um dado TTI. As restrições de QoS são apresentadas em [Lima et al. 2016], onde o principal objetivo foi a maximização da taxa total transmitida. Assim diferentemente de [Lima et al. 2016], neste artigo buscaremos uma maior eficiência energética através da alocação de potência adaptativa, a qual foi desconsiderada em [Lima et al. 2016]. Contudo, assumimos que a potência é igualmente distribuída entre os RBs assinalados a cada usuário. Essa consideração se faz necessária para que os os benefícios em termos de redução de PAPR não sejam perdidos. No entanto, usuários diferentes podem ajustar o nível de potência transmitida de acordo com seus objetivos.

Assim, assuma $\mathbf{X}$ como uma matriz binária com elementos $x_{j,p,m}$ que assume 1 se o j-ésimo usuário utilizando o p-ésimo padrão de assinalamento atingir o m-ésimo nível de MCS e 0 caso contrário. Note que $\mathbf{X}$ consiste na variável de otimização. De acordo com as considerações feitas anteriormente, o problema proposto é apresentado a seguir

$$\min_{\mathbf{X}} \left(\sum_{j \in \mathcal{J}} \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} x_{j,p,m} \cdot p_{j,p,m} \right), \tag{5a}$$

sujeito a

$$\sum_{j \in \mathcal{J}} \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} a_{n,p} \cdot x_{j,p,m} = 1, \ \forall n \in \mathcal{N}, \tag{5b}$$

$$\sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} x_{j,p,m} = 1, \ \forall j \ \in \ \mathcal{J}, \tag{5c}$$

$$\sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} p_{j,p,m} \cdot x_{j,p,m} \leq P_{T_j}, \ \forall j \ \in \ \mathcal{J}, \tag{5d}$$

$$x_{j,p,m} \in \{0, 1\}, \ \forall j \in \mathcal{J}, \ \forall p \in \mathcal{P} \text{ e } \forall m \in \mathcal{M}, \tag{5e}$$

$$\sum_{j \in \mathcal{J}_s} u \left(\sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} r_{j,p,m} \cdot x_{j,p,m}, \ t_j \right) \geq k_s, \ \forall s \in \mathcal{S}. \tag{5f}$$

Temos que a função objetivo mostrada em (5a) consiste na potência total transmitida pelos terminais. As restrições (5b) e (5e) asseguram que os RBs não serão reutilizados dentro da célula. As restrições (5c) e (5e) garantem que apenas um padrão de assinalamento seja adotado por cada usuário. Além disso, a restrição (5d) assegura que a potência usada por cada terminal j não exceda a máxima disponível. Por fim, a restrição (5f) estabelece que um número mínimo de usuários, k_s, deve ser satisfeito para cada serviço.

5. Caracterização da Solução Ótima

O problema (5) consiste em um problema de otimização combinatorial não linear devido a restrição (5f). Por conta da não linearidade do problema, a solução ótima só poderá ser encontrada por busca exaustiva, o que a torna sua obtenção impraticável mesmo para valores moderados de J, N, M e S.

De acordo com [Lima et al. 2016], o problema (5) pode ser reformulado para um problema de otimização linear inteiro, através da adição de novas variáveis de otimização e restrições. Assim, assuma ρ_j uma variável de seleção binária que assume 1 se o usuário j é selecionado para ser satisfeito e 0 caso contrário. Considerando essa nova variável, o problema (5) pode ser reformulado substituindo a restrição (5f) por duas novas restrições como segue

$$\sum_{p\in\mathcal{P}}\sum_{m\in\mathcal{M}} r_{j,p,m}\cdot x_{j,p,m} \geq \rho_j \cdot t_j,\ \forall j \in \mathcal{J}. \tag{6a}$$

$$\sum_{j\in\mathcal{J}} \rho_j \geq k_s,\ \forall s\ \in \mathcal{S}. \tag{6b}$$

Ao transformar o problema 5 em um problema ILP, podemos utilizar para sua resolução diversos *softwares* disponíveis no mercado que utilizam principalmente algoritmos baseados no método BB (do inglês, *Branch and Bound*) [Nemhauser and Wolsey 1988].

6. Estudo de Caso - Avaliação de Desempenho para Enlace Reverso do Sistema LTE

Esta seção é dedicada a análise de desempenho através de simulações computacionais da solução proposta para o novo problema apresentado neste artigo. Na seção 6.1, apresentamos em detalhes as principais hipóteses, modelos e parâmetros usados nas simulações. Na seção 6.2, apresentamos e discutimos os resultados obtidos.

6.1. Considerações para Simulação

Os principais aspectos de um sistema SC-FDMA no enlace reverso apresentado na seção 3 foram considerados no simulador computacional. Assume-se que um RB consiste de 12 subportadoras adjacentes no domínio da frequência e tem 1 ms de comprimento no domínio do tempo conforme consta nas especificações do sistema LTE [3GPP 2017].

O estado do canal é modelado pelos mecanismos de propagação mais importantes: modelo de perda de percurso dependente da distância, componente de sombreamento log-normal e componente de desvanecimento rápido com distribuição *Rayleigh*. Assumimos

Tabela 2. Principais Parâmetros para Simulação

Parâmetro	Valor	Unidade
Raio da célula	334	m
Potência disponível no terminal	24	dBm
Número de RBs	15	–
Número de MCS	15	–
Número de usuários	8	–
Número de subportadoras por RB	12	–
Desvio padrão sombreamento	8	dB
Perda de percurso	$35.3 + 37.6 \cdot \log_{10} d$	dB
Densidade espectral do ruído	$3.16 \cdot 10^{-20}$	W/Hz
Número de snapshots	3000	–
Requisitos de taxa de dados dos usuários	100 *a* 900	$kbps$

que a adaptação de enlace é realizada com base no relatório de 15 indicadores de qualidade de canal discretos ou CQIs (do inglês, *Channel Quality Indicators*) utilizados pelo sistema LTE [3GPP 2009]. Os limiares de SNRs para a comutação dos níveis de MCS foram obtidos por simulações de nível de enlace de [Mehlführer et al. 2009]. A potência total disponível nos terminais é de 24 dBm. A metodologia de simulação consiste em aplicar a solução proposta em diferentes realizações estatísticas (ou *snapshots*), tomando amostras diferentes das variáveis aleatórias que modelam o posicionamento do usuário e o estado do canal. Os principais parâmetros são mostrados na Tabela 2.

Para avaliar o desempenho da solução proposta para o problema aqui estudado consideramos um cenário em que o sistema disponibiliza dois serviços, ou seja $S = 2$, o qual tem um total de oitos usuários ativos, $J = 8$, distribuídos igualmente entre os serviços, $J_1 = 4$ e $J_2 = 4$. Assumimos também, que quatro usuários devem ser satisfeitos para o serviço 1, $k_1 = 4$, e 3 usuários satisfeitos para o serviço 2, $k_2 = 3$. Além disso, assumimos diferentes requisitos de taxa de dados dos usuários, t_j, conforme apresentado na Tabela 2.

Os algoritmos simulados nesse trabalho são:

- A solução do problema proposto por este trabalho na seção 5. Nas figuras, essa solução será identificada por Proposta.
- A solução do problema URM apresentado em [Lima et al. 2016] que consiste no problema de maximização da taxa de dados sem a restrição de QoS presente em (5f). Note que nesse caso os requisitos de taxa de dados e outras variáveis relacionadas a QoS não são críticas para o problema. A solução será identificada nas figuras como URM OPT.
- A solução do problema CRM apresentado em [Lima et al. 2016] que não utiliza alocação adaptativa de potência. Essa solução será denominada nas figuras como CRM OPT.

Utilizamos o *IBM ILOG CPLEX Optimizer* [ILOG 2014] para resolver os problemas de otimização linear inteiro. Para que haja uma comparação justa entre os algoritmos, as realizações do canal foram as mesmas para todos os algoritmos simulados. A escolha

dos valores para J, P, M e S é limitada pela complexidade computacional para obter as soluções ótimas.

Neste trabalho as principais métricas de desempenho utilizadas para avaliar os algortimos são:

- Potência total transmitida: Essa métrica está relacionada à função objetivo do problema (5). A potência total é a soma das potências transmitidas por todos os terminais em um dado TTI ou *snapshot*. O percentual de potência utilizada consiste na razão entre a soma das potências utilizadas por todos terminais móveis e a soma das potências máximas que poderiam ser transmitidas por todos terminais móveis, isto é, $J \cdot P_{T_j}$. O número de *snapshots* realizados nas simulações foi suficiente para assegurar um baixo desvio padrão nas estatísticas coletadas que ficaram entre 0.06% e 1.7%.
- Taxa de *outage*: Esta métrica está relacionada com as restrições mínimas de satisfação do problema (5). A taxa de *outage* é definida como a relação entre o número de *snapshots* com eventos de *outage* e o número total de *snapshots*. Um *outage* acontece quando uma solução específica não consegue encontrar uma solução viável, isto é, o algoritmo não encontra uma solução que cumpra as restrições do problema (5). A taxa de *outage* mostra a capacidade dos algoritmos em encontrar uma solução viável para o problema CRM.
- Taxa de dados média: Esta métrica está associada com as restrições mínimas de satisfação do problema (5). A taxa total de dados é a soma da taxa de dados de todos os terminais em um dado *snapshot*. A taxa de dados média é a razão entre a soma da taxa total de dados e o número de *snapshots* que não ocorre *outage*. As simulações realizaram um número de *snapshots* suficiente para garantir um baixo desvio padrão nas estatísticas coletadas que ficaram entre 2.39 e 89.02 kbits/s.

6.2. Resultados

Na Figura 1 temos a taxa de *outage* versus a taxa de dados requisitada por todos os usuários para a solução CRM OPT, URM OPT e para a solução proposta. Inicialmente, podemos observar que a taxa de *outage* aumenta à medida que a taxa de dados requisitada aumenta para todas as soluções. Isso já é algo esperado visto que o aumento da taxa requisitada torna a obtenção da satisfação do número mínimo de usuários mais difícil. Outra observação, também já esperada, consiste no fato da solução URM OPT apresentar altas taxas de *outage* mesmo nos casos em que se assume baixos valores de requisitos de taxas de dados. A razão para esse comportamento é que a solução URM OPT consiste na maximização da taxa de dados sem considerar os aspectos de QoS. Consequentemente, apenas os usuários com as condições de canais mais favoráveis obtêm a maior parte dos recursos, levando a uma quantidade de usuários satisfeitos inferior ao mínimo exigido pelo sistema. Além disso, também observamos que a taxa de *outage* da solução CRM OPT e da solução proposta são iguais para todas as taxas de dados requisitadas. Isso nos mostra que a solução proposta consegue atender os requisitos do sistema da mesma forma que a solução CRM OPT.

Na Figura 2 temos a taxa total de dados média versus a taxa de dados requisitada por cada terminal para as soluções CRM OPT, URM OPT e para a solução proposta. A principio, podemos ver que a taxa total de dados da solução proposta aumenta à medida

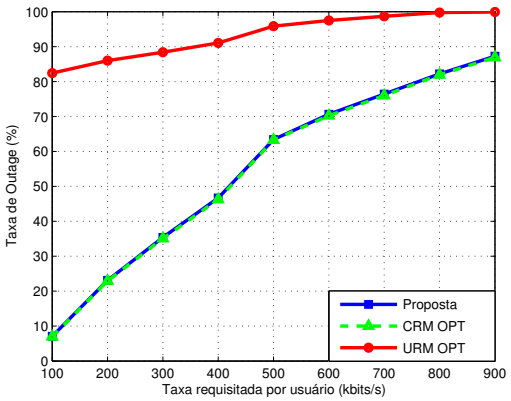


Figura 1. Taxa de *outage* para a solução CRM OPT, URM OPT e para a solução proposta.

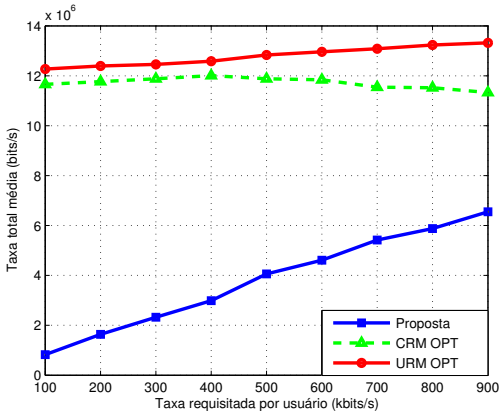


Figura 2. Taxa total de dados média para a solução CRM OPT, URM OPT e para a solução proposta.

que a taxa de dados requisitada aumenta. A razão para esse comportamento já era esperada visto que a solução proposta busca apenas satisfazer a taxa requisitada de cada usuário de forma a economizar a potência total transmitida. Ainda de acordo com a Figura 2, ao compararmos os três algoritmos podemos observar que a solução URM OPT e a solução CRM OPT apresentam as maiores taxas de dados. Este resultado deve-se ao fato destes algoritmos terem por objetivo a maximização da taxa total transmitida no sistema.

Uma análise descompromissada dos resultados apresentados nas Figuras 1 e 2 podem levar a uma conclusão errônea quanto a relevância da solução proposta neste artigo. Isso deve-se ao fato da solução CRM OPT obter baixas taxas de outage e ao mesmo tempo altíssimas taxas de transmissão, enquanto que as taxas apresentadas pela solução proposta neste artigo são mais moderadas. Contudo, a análise da Figura 3 mostra um aspecto importante a ser considerado nesta análise. Na Figura 3 apresentamos o percentual de potência utilizada versus a taxa de dados requisitada por cada usuário das soluções CRM OPT, URM OPT e da solução proposta. Primeiramente, podemos observar que o percentual de potência utilizada aumenta conforme aumentamos a taxa de dados requisitadas pelos usuários para a solução proposta. Isto é esperado visto que quanto maiores

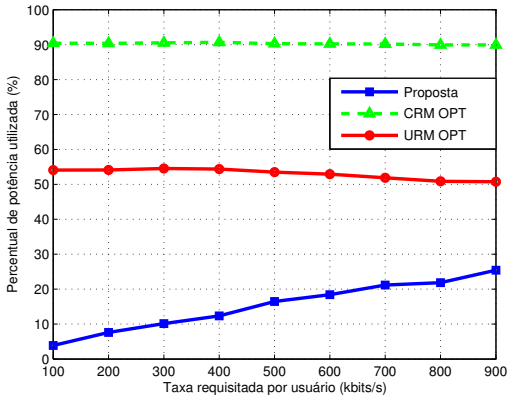


Figura 3. Percentual de potência utilizada pela solução CRM OPT, URM OPT e pela solução proposta.

são as demandas dos usuários em termos de QoS, maiores são as dificuldades em satisfazer as restrições do problema estudado, consequentemente, uma maior potência de transmissão é requisitada. As soluções URM OPT e CRM OPT em geral permitem que seus terminais utilizem a máxima potência de transmissão. P_{T_j}. Além disso, comparando o desempenho relativo dos algoritmos na Figura 3 observamos que a solução URM OPT apresenta uma maior economia em termos de potência quando comparada a solução CRM OPT. Porém, conforme visto anteriormente, a solução URM OPT não é capaz de satisfazer o QoS dos usuários conforme visto na Figura 1. Como resultado mais importante, destacamos a grande economia de potência realizada pela solução proposta comparada a solução CRM OPT. Considerando uma taxa de dados requisitada de 500 kbps, podemos ver que a solução proposta apresenta um consumo de aproximadamente 16% da potência total disponível, enquanto a soluções CRM OPT e URM OPT apresentam um consumo de aproximadamente 90% e 53%, respectivamente.

Uma análise conjunta das Figuras 1, 2 e 3 nos permite concluir que a solução proposta é capaz de satisfazer o QoS e o número mínimo de usuários satisfeitos com uma quantidade de potência utilizada bem menor que a utilizada pelas outras soluções. Apesar da solução proposta apresentar uma menor taxa total transmitida quando comparada a solução CRM OPT na Figura 2, essa menor taxa total transmitida não compromete a satisfação dos usuários. Uma menor quantidade de potência transmitida conforme mostrada na Figura 3, traz uma série de vantagens em sistemas celulares devido a diminuição drástica da interferência intercelular gerada no sistema. Além disso, consumo de energia racional consiste em um dos pilares no projeto das redes modernas.

7. Conclusões e Perspectivas Futuras

Neste trabalho, estudamos o problema de minimização da potência total transmitida sujeito a restrições de satisfação de QoS e com alocação de potência adaptativa no enlace reverso do sistema LTE empregando SC-FDMA. O sistema SC-FDMA exige que os RBs atribuídos a um determinado usuário sejam adjacentes uns aos outros a fim de assegurar uma baixa PAPR.

Mostramos que o problema formulado consiste em um problema de otimização combinatorial não linear. Em virtude da alta complexividades desses tipos de proble-

mas, convertemos o mesmo em um problema inteiro linear por meio da adição de novas variáveis de otimização e restrições. Esse tipo de problema, apesar da complexidade exponencial, pode ser resolvidos por técnicas convencionais. Os resultados das simulações computacionais mostram que a solução proposta obteve resultados expressivos em termos de eficiência energética quando comparada as demais soluções sem comprometer a satisfação com o QoS requisitado pelos usuários. Os ganhos de economia de potência são superiores a 70% quando comparado a solução CRM OPT.

Por fim, diante dos bons resultados obtidos, como perspectivas deste trabalho, podemos mencionar o desenvolvimento de soluções subótimas com menor complexidade do que a solução proposta, assim como considerar outros requisitos de QoS ou QoE (do inglês, *Quality of Experience*) que levam em conta a latência na transmissão de pacotes, por exemplo.

Agradecimentos

Este trabalho foi financiado pelo Centro de Inovação da Ericsson Telecomunicações S.A., Brasil, sob o contrato de cooperação técnico EDB/UFC.43. O estudante Iran M. B. Júnior agradece a Coordenação de Acompanhamento de Discente (CAD) da UFC pelo suporte financeiro concedido por meio do Programa de Educação Tutorial (PET) e F. Rafael M. Lima agradece a Fundação Cearense de Apoio a Pesquisa (FUNCAP) pelo apoio financeiro em forma de bolsa de produtividade em pesquisa.

Referências

3GPP (2009). Evolved universal terrestrial radio access (E-UTRA); physical layer procedures. Third Generation Partnership Project, Tech. Rep. TR 36.213 V8.6.0, Mar. 2009.

3GPP (2017). 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2 (Release 14). Technical Specification, 36.300 V14.2.0, Mar. 2017.

Capozzi, F., Piro, G., Grieco, L. A., Boggia, G., and Camarda, P. (2013). Downlink packet scheduling in LTE cellular networks: Key design issues and a survey. *IEEE Communications Surveys & Tutorials*, 15(2):678–700.

Delgado, O. and Jaumard, B. (2010). Scheduling and resource allocation for multiclass services in LTE uplink systems. In *2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications*, pages 355–360. IEEE.

Furuskär, A. (2003). Radio resource sharing and bearer service allocation for multi-bearer service, multi-access wireless networks.

ILOG, I. (2014). Cplex optimization studio. *URL: http://www-01. ibm. com/software/commerce/optimization/cplex-optimizer*.

Jang, J. and Lee, K. B. (2003). Transmit power adaptation for multiuser OFDM systems. *IEEE Journal on selected areas in communications*, 21(2):171–178.

Lim, J., Myung, H. G., Oh, K., and Goodman, D. J. (2006). Channel-dependent scheduling of uplink single carrier FDMA systems. In *IEEE Vehicular technology conference*, pages 1–5. IEEE.

Lima, F. R. M., Maciel, T. F., and Cavalcanti, F. R. P. (2016). Radio resource allocation in SC-FDMA uplink with resource adjacency constraints. *Journal of Communication and Information Systems*, 31(1).

Lima, F. R. M., Wänstedt, S., Cavalcanti, F. R. P., and Freitas, W. C. (2010). Scheduling for improving system capacity in multiservice 3GPP LTE. *Journal of Electrical and Computer Engineering*, 2010:3.

Maurício, W. V., Lima, F. R. M., Maciel, T. F., and Cavalcanti, F. R. P. (2014). Spectral and energy efficiency with satisfaction constraints. In *Telecommunications Symposium (ITS), 2014 International*, pages 1–5. IEEE.

Mehlführer, C., Wrulich, M., Ikuno, J. C., Bosanska, D., and Rupp, M. (2009). Simulating the long term evolution physical layer. In *Signal Processing Conference, 2009 17th European*, pages 1471–1478. IEEE.

Myung, H. G., Lim, J., and Goodman, D. J. (2006). Single carrier FDMA for uplink wireless transmission. *IEEE Vehicular Technology Magazine*, 1(3):30–38.

Nemhauser, G. L. and Wolsey, L. A. (1988). Integer programming and combinatorial optimization. *Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin*, 20:8–12.

Nwamadi, O., Zhu, X., and Nandi, A. (2008). Dynamic subcarrier allocation for single carrier-FDMA systems. In *Signal Processing Conference, 2008 16th European*, pages 1–5. IEEE.

Ruby, R. and Leung, V. (2011). Towards QoS assurance with revenue maximization of LTE uplink scheduling. In *Communication Networks and Services Research Conference (CNSR), 2011 Ninth Annual*, pages 202–209. IEEE.

Seol, C., Cheun, K., and Hong, S. (2010). A statistical inter-cell interference model for downlink cellular OFDMA networks under log-normal shadowing with ricean fading. *IEEE Communications Letters*, 14(11):1011–1013.

Shi, T., Zhou, S., and Yao, Y. (2004). Capacity of single carrier systems with frequency-domain equalization. In *Emerging Technologies: Frontiers of Mobile and Wireless Communication, 2004. Proceedings of the IEEE 6th Circuits and Systems Symposium on*, volume 2, pages 429–432. IEEE.

Triantafyllopoulou, D., Kollias, K., and Moessner, K. (2015). QoS and energy efficient resource allocation in uplink SC-FDMA systems. *IEEE Transactions on Wireless Communications*, 14(6):3033–3045.

Wong, C. Y., Cheng, R. S., Letaief, K. B., and Murch, R. D. (1999). Multiuser OFDM with adaptive subcarrier, bit, and power allocation. *IEEE Journal on selected areas in communications*, 17(10):1747–1758.

Wong, I. C., Oteri, O., and McCoy, W. (2009). Optimal resource allocation in uplink SC-FDMA systems. *IEEE Transactions on Wireless Communications*, 8(5):2161–2165.

Wu, G., Yang, C., Li, S., and Li, G. Y. (2015). Recent advances in energy-efficient networks and their application in 5G systems. *IEEE Wireless Communications*, 22(2):145–151.

Uma Análise da Evolução e Características de Falhas em uma Rede de Telecomunicações de Médio Porte

Leandro A. de Sá Vieira e Ítalo Cunha

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais

{lvieira,cunha}@dcc.ufmg.br

Abstract. *The growth of the Internet and increasing demand for networked services require continued advances in communication technologies. Unfortunately, larger and more complex networks also suffer a higher number of incidents. Even though network incidents have significant negative impact on individuals and businesses, incident causes and characteristics are still poorly understood. In particular, network operators consider information about network incidents sensitive and private. In this work we analyze a dataset of network incidents on a regional Brazilian transit network with presence in five states. Our results further our understanding of transit network incidents, particularly as a function of network evolution. As an example, we find that the transition from circuit switching to packet switching technology led to an* increase *in the rate of incidents but also to a* decrease *in time to incident resolution.*

Resumo. *O crescimento da Internet e da demanda por serviços em rede dependem de avanço contínuo das tecnologias de comunicação. Porém, o crescimento da rede e de sua complexidade resultam em aumento do número de incidentes. Apesar do grande impacto de incidentes de rede no cotidiano de pessoas e empresas, nosso entendimento sobre suas causas e características ainda é muito limitado. Em particular, provedores consideram sensíveis e sigilosas informações sobre incidentes em suas redes e não divulgam estas informações. Neste trabalho analisamos um conjunto de dados contendo informações sobre incidentes na rede de uma operadora de telecomunicações de médio porte com atuação em cinco estados do Brasil. Nossos resultados melhoram nosso entendimento sobre incidentes de rede, especialmente em função da evolução da rede. Por exemplo, encontramos que a transição de sistemas de comutação por circuitos para sistemas de comutação por pacotes levou a um* aumento *da taxa de incidentes mas também a uma* redução *no tempo de resolução.*

1. Introdução

Conceber uma rede de telecomunicações imune a incidentes adversos é impossível. Diante disso, um melhor entendimento de incidentes—como frequência, características e causas—é essencial para instruir e aprimorar a gerência de redes [Govindan et al. 2016]. Melhor entendimento de incidentes permite a criação de novos procedimentos de verificação (do *software* e sua configuração, bem como do *hardware*)

[Gember-Jacobson et al. 2016, Beckett et al. 2016, Shaikh and Greenberg 2004], aprimoramento de rotinas para identificação das causas de incidentes [Dhamdhere et al. 2007, Nguyen et al. 2009, Kompella et al. 2005], implementação de ferramentas de monitoramento com maior cobertura e diagnóstico mais preciso [Mahimkar et al. 2008, Sommers et al. 2007, Cunha et al. 2009], ou desenvolvimento de mecanismos para contornar ou mitigar o impacto de incidentes [Peter et al. 2014, Katz-Bassett et al. 2012]. Infelizmente, muito poucos dados sobre incidentes de rede estão disponíveis publicamente, o que limita nossa capacidade de estudá-los. Isto acontece por que operadoras de telecomunicações consideram estas informações segredos de negócio sensíveis [Markopoulou et al. 2008, Dainotti et al. 2011, Turner et al. 2010].

Operadoras de telecomunicações investem esforço significativo para solucionar os inevitáveis incidentes de rede e garantir disponibilidade (*uptime*) da rede. Um relatório da Cisco indica que recursos humanos são 50% do custo de operação de uma rede [Cisco 2011]. Centros de gerência de rede empregam técnicos responsáveis por atividades como a conexão de novos clientes, expansão da infra-estrutura e configuração de dispositivos. Além disso, os centros de gerência também empregam plantão 24/7 para resposta a incidentes. Centros de gerência utilizam sistemas de rastreamento de incidentes (*issue tracking*) para coordenar a ação dos técnicos.

Neste trabalho caracterizamos dezenas de milhares de registros de incidentes no sistema de rastreamento do centro de gerência de uma operadora de telecomunicações com atuação em cinco estados no Brasil. Os registros cobrem incidentes que ocorreram entre janeiro de 2009 e setembro de 2016.

Nós avaliamos os registros de incidentes em quatro dimensões: tecnologia, causa do incidente, localização geográfica e ao longo do tempo. Quanto à tecnologia, contrastamos incidentes em dispositivos de comutação por circuitos (PDH/SDH) e incidentes em dispositivos de comutação de pacotes (Metro Ethernet e GPON-FTTH). Quanto à causa do incidente, consideramos causas como erros de configuração ou rompimento de fibra óptica. Quanto à localização geográfica, estudamos características de incidentes em capitais e no interior. E quanto ao tempo, o longo período de cobertura dos registros nos permite estudar características dos incidentes em função da evolução da rede. Caracterizamos os incidentes principalmente em função dos componentes do tempo de resolução (e.g., tempo de diagnóstico, tempo de análise e tempo para alocação de equipe de campo) e do tempo entre ocorrências de incidentes.

Nossos resultados melhoram nosso entendimento do impacto da evolução de uma rede sobre falhas e do impacto das falhas sobre o desempenho da rede. Por exemplo, encontramos que a frequência de incidentes em um enlace diminui à medida que seu tempo em produção aumenta; que a tecnologia de um enlace impacta a frequência de incidentes e a necessidade de intervenção em campo; que algumas causas de incidentes não são bem identificadas; e que enlaces no interior apresentam falhas com frequência maior que enlaces em capitais.

Nossos resultados podem direcionar esforços de operadores para tentar reduzir o tempo e custo de resolução de incidentes ou ajudar pesquisadores identificarem novos desafios de gerenciamento em redes modernas.

2. Estrutura da Rede

Infra-estrutura física. A rede analisada tem extensa infra-estrutura em um estado do Brasil, e mantém presença também em outros quatro estados vizinhos. A rede possui equipamentos em mais de 1000 localidades distintas. Interconexões entre PoPs do *backbone* são realizadas em topologia de anel, o que fornece certa redundância contra particionamento da rede. O *backbone* utiliza enlaces de comutação por circuito com taxas de transmissão a partir a 155 Mbps e enlaces de comutação por pacotes com taxas de transmissão a partir de 1 Gbps. A grande maioria dos enlaces entre PoPs utilizam fibra óptica e uma minoria utiliza radiofrequência. A infra-estrutura dos PoPs varia, mas geralmente inclui sistema de refrigeração, transformadores de corrente alternada em corrente contínua, sistema de energia sobressalente (baterias ou geradores a combustão), distribuidor óptico para organização do cabeamento e racks acomodando os equipamentos.

A rede tem capacidade agregada para aproximadamente 250 Gbps de tráfego externo, sendo 50 Gbps em três pontos de troca de tráfego (PTT) nacionais, 40 Gbps de trânsito internacional, e 160 Gbps em enlaces ponto-a-ponto comerciais. A quantidade e capacidade de equipamentos provisionados em um PoP são provisionados de acordo com a densidade de clientes atendidos e o volume de tráfego no PoP. Os clientes geralmente são conectados aos PoPs utilizando topologia estrela.

Gerenciamento da rede. O centro de gerência de redes (CGR) emprega 21 técnicos responsáveis por monitorar e realizar atividades corretivas 24/7. O CGR realiza monitoração pró-ativa na infra-estrutura dos PoPs através de sensores que reportam anomalias físicas, como aumento de temperatura e falta de energia. O monitoramento da rede possui um maior foco nos enlaces de *backbone*, onde incidentes têm impacto muito mais significativo. O monitoramento da rede é realizado através de ferramentas que detectam eventos como quedas de links, perda de pacotes e congestionamentos.

Um incidente pode ser detectado por processos de monitoramento ou informado através de contato realizado por um cliente. Após a notificação de um incidente, ele é tratado segundo o fluxograma na figura 1. Um técnico do centro de gerência analisa o incidente para tentar identificar o problema. Se um técnico não conseguir identificar o problema, outro técnico é atribuído para analisar o incidente. Após identificado o problema, um técnico decide se é necessário intervenção local via equipe de campo ou se o incidente pode ser solucionado remotamente.[1] Caso seja necessária intervenção local, uma equipe de campo é escalada e despachada. Após intervenção local, a falha é reavaliada pelo centro de gerência.

As equipes de campo são geograficamente distribuídas em localidades estratégicas, trabalhando em escalas de revezamento mantendo disponibilidade para acionamento 24/7. As capitais concentram cerca de 40% das equipes de campo. Geralmente cada equipe é formada por técnicos de equipamentos, responsáveis por atendimento nas instalações de clientes e PoPs, técnicos de fibra óptica, responsáveis pela manutenção na rede óptica, e técnicos de trabalho pesado, responsáveis por serviços como o lançamento de cabos entre postes.

[1]Comutadores e roteadores de propriedade da operadora instalados nos clientes podem ser acessados remotamente através de uma VLAN de gerência. Aproximadamente 90% dos equipamentos legados como modems PDH e conversores de mídia permitem gerencia remota.

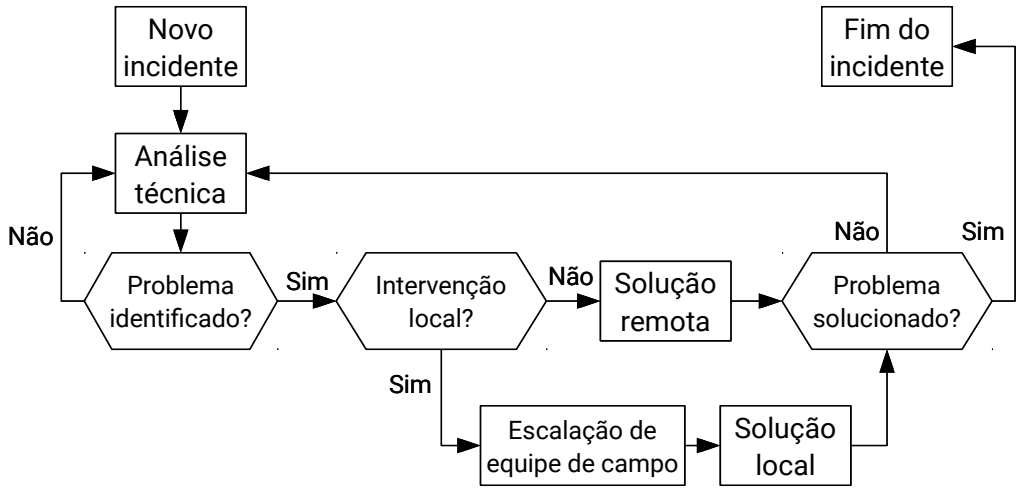


Figura 1. Fluxograma do tratamento de um incidente.

3. Conjunto de dados

Para cada incidente de rede identificado é aberto um registro num sistema de rastreamento de incidentes onde são inseridas informações sobre o tratamento do incidente até sua resolução. Nosso conjunto de dados contém dezenas de milhares de registros de falhas cobrindo sete anos de operação da rede, de janeiro de 2009 até setembro de 2016. A maioria (85%) dos registros foram originados a partir de reclamações realizadas pelos clientes e 15% foram originados de monitoramento pró-ativo da rede. Os dados são considerados sensíveis pela operadora e privados. Similar a outros trabalhos, acordamos com a operadora de focar nossos resultados em uma análise relativa dos incidentes de rede.

O primeiro técnico atribuído a um incidente é do centro de gerência e responsável pela análise inicial. A análise de um incidente pode levar à identificação do problema ou à atribuição do incidente a outro técnico que irá continuar a análise. Cada vez que um técnico é atribuído a um incidente, seu nome, equipe e horário de atribuição são registrados no sistema. Cada vez que um técnico termina sua análise, o horário do término também é registrado no sistema.

Como os registros são preenchidos manualmente, as informações inseridas podem conter erros ou imprecisões. Para reduzir o impacto de erros de preenchimento em nossa análise, agregamos períodos de tempo e desconsideramos o tempo de análise de cada técnico atribuído a um incidente. Definimos o *tempo de análise* de um incidente como o período entre os instantes de abertura do registro do incidente e de identificação do problema. Definimos o *tempo de solução* de um incidente como o período entre os instantes de abertura e de fechamento do registro. (Algumas vezes a equipe de campo acionada continua trabalhando *in loco* após o fechamento de um registro; não contabilizamos este período no tempo de solução do incidente.) Também filtramos registros de incidentes incompatíveis com o fluxograma mostrado na figura 1. Em particular, filtramos 3,5% de registros de incidentes nos quais a equipe solucionadora foi acionada antes da abertura do incidente. Filtramos também 38% dos incidentes cuja causa no sistema estava marcada como "não procedente": em sua maior parte incidentes reportados por clientes cuja causa

CAUSA	EXEMPLO DE DESCRIÇÃO
Configuração	"IP configurado erroneamente", "Cross conexão refeita"
Hardware	"Mau contato no cabo UTP", "Defeito na interface óptica"
Energia	"Falta de energia comercial", "Defeito na fonte de alimentação"
Rompimento	"Cabo óptico rompido", "Cabo óptico atenuado"
Outros	"Normalizou sem intervenção", "Falha na operadora parceira"

Tabela 1. Classificação de causas de incidentes.

não está na rede da operadora (e.g., incidentes na rede do próprio cliente).

Um registro de incidente contém ainda campos explícitos para identificação do dispositivo onde o incidente foi detectado, uma descrição da causa do incidente, localização geográfica e equipes envolvidas no solucionamento. Utilizamos estes registros para analisar os incidentes sob diferentes pontos de vista. Os registros possuem também um campo para descrição detalhada da causa e ações tomadas para solucionamento. Utilizamos este campo para melhor entender os incidentes.

4. Método de análise

Classificaçao dos incidentes. Nós classificamos os incidentes em quatro dimensões distintas. A primeira dimensão classifica incidentes por *tipo de tecnologia do enlace*, que pode ser de três tipos: enlaces de comutação por circuito PDH/SDH, utilizado em conexões entre PoPs e em serviços de *last mile* providos para outras operadoras; enlaces de comutação de pacotes Metro Ethernet, utilizado em conexão entre PoP e no acesso de clientes; e enlaces GPON-FTTP (*fiber to the premises*) que atende uma área residencial com internet, voz e TV a cabo.

A segunda dimensão classifica incidentes por *localização geográfica*: consideramos que incidentes podem ocorrer na capital ou no interior do estado onde a operadora concentra sua infra-estrutura. A terceira dimensão classifica incidentes por *tipo de causa*; classificamos os 251 identificadores de causa de incidente no sistema de rastreamento nas cinco classes mostradas na tabela 1. A tabela mostra alguns exemplos de identificadores de causa associados a cada classe. Por último, na quarta dimensão classificamos incidentes por *ano de ocorrência*.

Características dos incidentes. Caracterizamos três métricas principais dos incidentes: o tempo de solução (tempo de abertura e fechamento de um registro, seção 3), o tempo de reparo em campo e o tempo entre incidentes. Quando uma equipe solucionadora é acionada, um técnico analisa o incidente para escalar uma equipe de campo adequada. Nós descartamos esse tempo de análise; assim, o *tempo de reparo em campo* considera apenas as atividades realizadas exclusivamente por equipes de campo. O *tempo entre incidentes* é calculado por enlace, subtraindo o tempo de abertura de cada par de registros de incidentes concecutivos no mesmo enlace. Observamos que cerca de 3% dos pares de incidentes consecutivos ocorriam no mesmo dia e, na maioria dos casos que analisamos manualmente, eram casos de duplicidade no sistema. Em nossa análise consideramos apenas um registro de incidente por dia em cada enlace.

5. Resultados

Nesta seção apresentamos os resultados da análise dos registros de incidente. Mostramos uma visão geral da frequência e distribuição dos incidentes (seção 5.1) e analisamos os incidentes sob cada uma das dimensões definidas na seção 4 (seções 5.2–5.4).

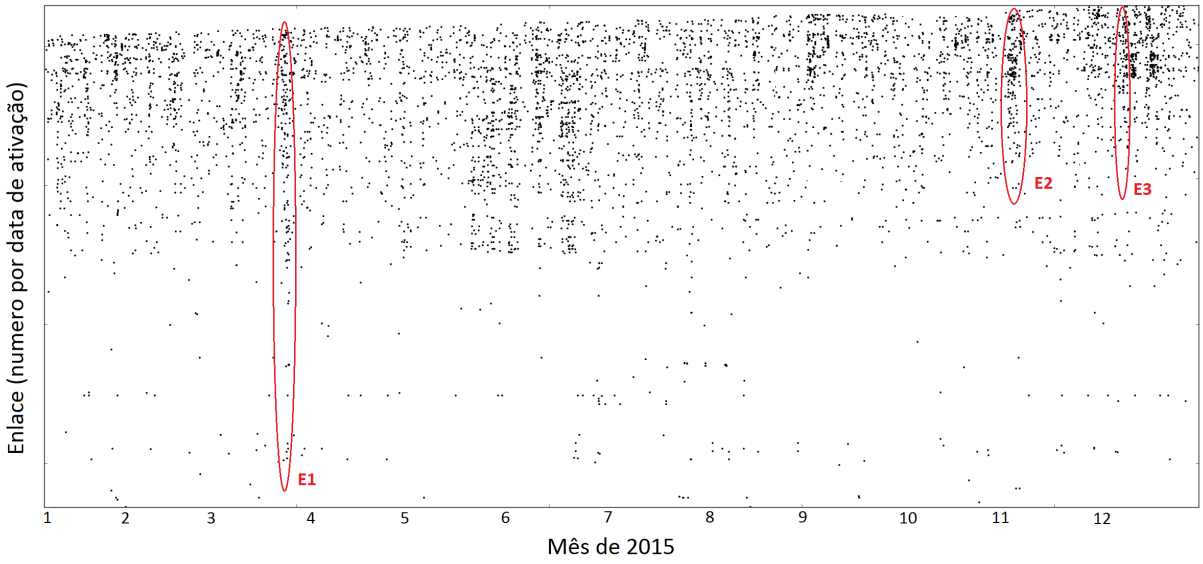


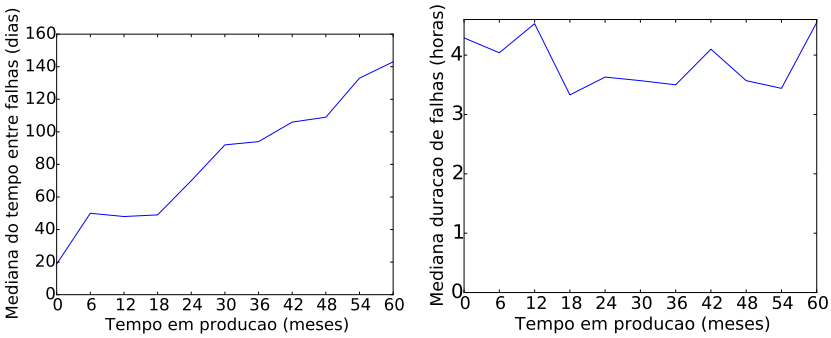
Figura 2. Visão geral dos incidentes de rede registrados em 2015. Os enlaces são ordenados ao longo do eixo Y pela data de ativação. Cada ponto representa o instante de abertura de um registro de incidente de rede.

5.1. Visão geral dos incidentes

A figura 2 mostra a visão geral dos incidentes de rede no ano de 2015. O eixo x mostra o tempo, o eixo y mostra os enlaces e cada ponto mostra o instante de abertura de um registro de incidente. Os enlaces estão ordenados ao longo do eixo y pela sua data de ativação (por isso o triangulo sem incidentes no canto superior esquerdo).

Mostramos os dados de 2015 pois em 2015 ocorreram 71% dos eventos que afetaram mais de 100 enlaces em nosso conjunto de dados. Resultados para outros anos são qualitativamente similares. Identificamos três dos eventos que afetaram mais de 100 enlaces em 2015 na figura 2 com círculos vermelhos.

O evento E1 resultou em incidentes em mais de 110 enlaces. O problema foi um duplo rompimento óptico que tirou de operação os enlaces principal e de redundância de um anel 10Gbps que conecta a capital com o interior do estado. No evento E2 temos representado um ciclo na rede de comutadores camada 2. Esse evento atingiu 151 enlaces de acesso de clientes, que ficaram inoperantes por duas horas, e foi causado por um erro de configuração. O evento E3 deixou 124 enlaces inoperantes por uma hora e meia e foi causado por outro ciclo na rede de comutadores camada 2 provocado por mal funcionamento dos comutadores de um anel 20Gbps. Para solucionar os incidentes foi necessário abrir um lado do anel e posteriormente atualizar o *firmware* de todos os comutadores envolvidos.



(a) Tempo entre incidentes (b) Tempo de resolução

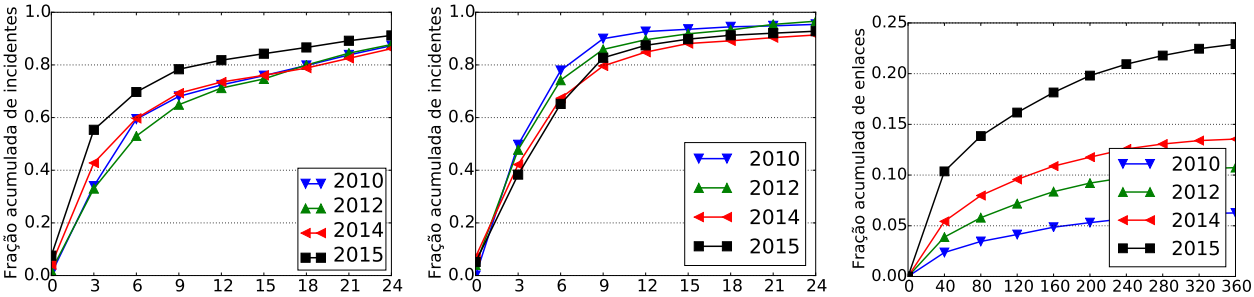
Figura 3. Características de incidentes em função do tempo que o enlace está em produção.

A figura 2 é similar a resultados anteriores caracterizando falhas em redes de trânsito [Markopoulou et al. 2008, Turner et al. 2010]. A figura mostra bandas verticais resultantes de incidentes ou atividades de manutenção que afetam vários enlaces. A figura mostra também (pequenas) bandas horizontais resultantes de enlaces com incidentes recorrentes.

A figura 2 mostra ainda que incidentes se concentram em enlaces ativados mais recentemente. A figura 3(a) confirma e quantifica essa observação mostrando a mediana do tempo entre falhas em função do tempo em produção (i.e., tempo desde a ativação de um enlace). A figura 3(b) mostra a mediana do tempo de resolução de incidentes em função do tempo em produção. Apesar dos enlaces em produção a menos tempo apresentarem maior frequência de incidentes, vemos que o tempo de resolução é pouco afetado pelo tempo em produção.

5.2. Evolução de incidentes ao longo do tempo

A figura 4(a) mostra o tempo de solução de falhas por ano. O tempo médio de solução de incidentes permaneceu estável entre 2009 e 2014, com tempo de solução médio mínimo de 9,3 horas (em 2010) e máximo de 10,0 horas (em 2013). Porém, percebemos uma significativa redução em 2015, com tempo médio de solução de 6,8 horas. Esta redução deve-se a um aumento significativo da fração de falhas que ocorrem em enlaces do tipo Metro Ethernet. Como mostraremos na seção 5.3, enlaces Metro Ethernet possuem tempo



(a) Tempo de solução (h) (b) Tempo de trabalho em campo (h) (c) Tempo entre incidentes (dias)

Figura 4. Análise de incidentes por ano.

Tabela 2. Evolução do perfil da infra-estrutura. Dados mostrados em escala relativa para para manter em sigilo os valores absolutos.

Ano	PDH/SDH		Metro Ethernet		GPON	
	Enlaces	Incidentes	Enlaces	Incidentes	Enlaces	Incidentes
2009	96,6%	94,2%	3,4%	5,8%	0,0%	0,0%
2010	92,3%	88,4%	4,6%	7,6%	3,0%	3,9%
2011	89,4%	75,5%	5,8%	19,7%	4,7%	4,8%
2012	85,7%	51,3%	6,7%	41,3%	7,5%	7,3%
2013	81,6%	48,7%	9,3%	40,7%	9,0%	10,6%
2014	76,5%	33,2%	15,2%	57,5%	8,2%	9,2%
2015	72,9%	22,0%	18,3%	68,8%	8,8%	9,2%
2016	68,7%	14,8%	22,5%	77,0%	8,8%	8,1%

de solução de incidente menor que as outras tecnologias empregadas pela operadora.

Apesar da aparente melhora no tempo de solução de incidentes em 2015, a figura 4(b) mostra que o tempo de trabalho em campo não apresenta melhora ao longo dos anos. Pelo contrário, vemos um pequeno aumento do tempo de trabalho em campo, o que pode ser resultado de expansão geográfica da rede ao longo dos anos e do aumento do número de clientes, o que exige mais deslocamentos das equipes de campo.[2]

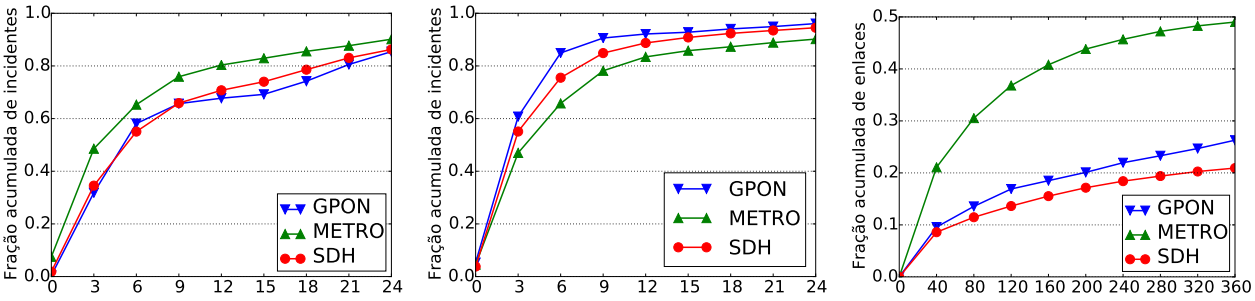
As longas caudas das distribuições nas figuras 4(a) e (b) devem-se geralmente a interrupções no trabalho de solução do incidente devido a fatores externos como a falta de segurança para o trabalho da equipe de campo, falta de autorização para atividade de manutenção em instalações de clientes, ou espera por equipamento.

A figura 4(c) mostra a distribuição do tempo entre incidentes nos enlaces. Calculamos o período entre pares de incidentes consecutivos em cada enlace. Contabilizamos cada par no ano de ocorrência do segundo incidente no par (o mais recente). Cortamos o eixo y em 0,25 para melhorar a legibilidade. As distribuições não chegam a 1,0 por que uma fração significativa dos enlaces não sofrem nenhum incidente; contabilizamos estes enlaces no gráfico com um tempo entre incidentes infinito. Percebemos uma clara redução do tempo entre incidentes, um resultado negativo. Novamente, este resultado deve-se ao aumento da fração de enlaces Metro Ethernet, que possui maior taxa de incidentes (seção 5.3).

5.3. Características de incidentes por tipo de tecnologia de enlace

O perfil da infra-estrutura de rede da operadora mudou ao longo dos anos em função de avanços tecnológicos e aumento do número de clientes. A tabela 2 mostra a fração de enlaces da rede de cada tipo das três tecnologias utilizadas. Observamos uma substituição gradativa dos enlaces PDH/SDH por enlaces Metro Ethernet, e o crescimento de enlaces GPON. A tabela 2 também mostra a fração de incidentes para cada tipo de tecnologia. Vemos que os enlaces Metro Ethernet apresentam incidentes muito mais frequentemente do que enlaces PDH/SDH e GPON.

[2]Os registros de incidente não contabilizam o tempo de deslocamento. Alternativamente, tentamos quantificar o deslocamento da equipe de campo (e.g., via consumo de combustível) para embasar essa hipótese mas não conseguimos obter esta informação.



(a) Tempo de solução (h) (b) Tempo de trabalho em campo (h) (c) Tempo entre incidentes (dias)

Figura 5. Análise de incidentes por tecnologia.

A figura 5 é análoga à figura 4, porém particiona os incidentes por tecnologia do enlace em vez de por ano de ocorrência. A figura 5(a) mostra que enlaces Metro Ethernet têm os menores tempos de solução de incidentes. Este resultado deve-se ao fato que incidentes em enlaces Metro Ethernet são resolvidos remotamente, sem intervenção via equipe de campo, 44% das vezes; incidentes em enlaces PDH/SDH e GPON podem ser resolvidos remotamente apenas 9% e 11% das vezes, respectivamente.

A figura 5(b) mostra a distribuição do tempo de trabalho das equipes de campo. Enlaces GPON têm menor tempo de trabalho em campo, provavelmente por que estes enlaces se concentram em regiões específicas onde existem planos de alta velocidade usando vibra óptica, o que reduz o tempo de deslocamento e heterogeneidade dos dispositivos. Apesar da menor necessidade de intervenção em campo, incidentes em enlaces Metro Ethernet em geral possuem maior tempo de resolução em campo. Um fator é que as intervenções em campo podem ser feitas em instalações de clientes, sobre as quais a operadora não tem controle.

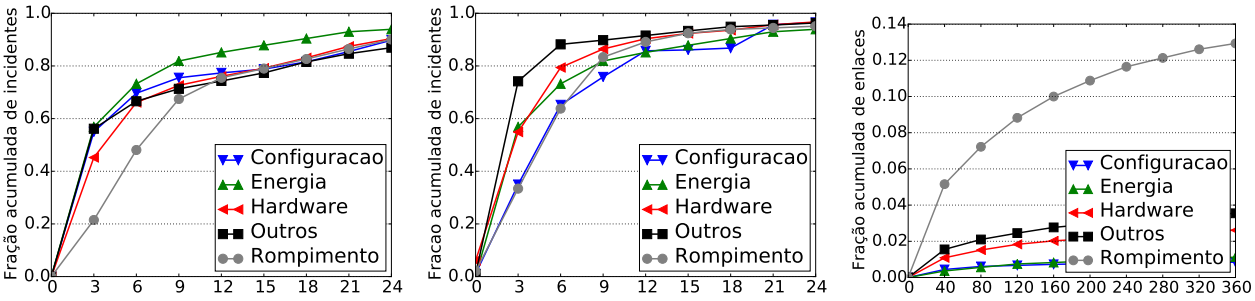
Por último, a figura 5(c) mostra a distribuição do tempo entre incidentes por enlace para cada uma das tecnologias de enlace. Note que o eixo y vai apenas até 0,5. A figura 5(c) confirma os resultados na tabela 2, confirmando que enlaces Metro Ethernet apresentam tempo entre falhas significativamente menor que enlaces PDH/SDH e GPON.

5.4. Características de incidentes por causa

A tabela 3 mostra a proporção de cada causa de incidente entre as tecnologias utilizadas. Vemos que a proporção de causas de incidentes é relativamente parecida entre todas as tecnologias utilizadas. Chamamos atenção para uma maior concentração de incidentes de configuração em enlaces Metro Ethernet e maior concentração de incidentes de *hardware* em enlaces GPON.

Tabela 3. Quantidade de incidentes por causa e tecnologia. Probabilidade de intervenção em campo em função da causa do incidente.

	PDH/SDH	Metro Ethernet	GPON	Intervenção em campo
Hardware	24.9%	22.4%	39.1%	80.6%
Configuração	4.0%	7.5%	5.3%	65.3%
Energia	7.7%	5.5%	1.1%	46.3%
Rompimento	47.0%	52.0%	52.6%	100.0%
Outros	16.3%	12.3%	3.1%	37.4%



(a) Tempo de solução (h) (b) Tempo de trabalho em campo (h) (c) Tempo entre incidentes (dias)

Figura 6. Análise de incidentes por causa de falhas.

A figura 6(a) mostra que os incidentes de rompimento possuem os maiores tempos de solução. Isto é esperado, visto a necessidade de intervenção de equipe de campo e necessidade de realizar fusões de fibra óptica. Incidentes de energia e configuração em geral apresentam menor tempo de resolução e as menores taxas de intervenção em campo.

Porém, como mostrado na figura 6(b), quando há necessidade de envio de equipe de campo na solução de incidentes de configuração, o tempo de trabalho em campo é semelhante ao de incidentes de rompimento. Este resultado ilustra a complexidade da configuração de redes modernas e a necessidade de modernos mecanismos de verificação e auxílio à gerência.

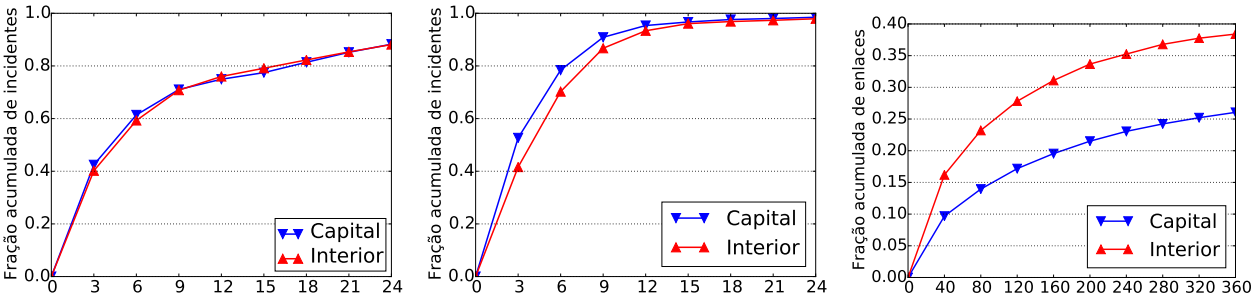
A figura 6(c) mostra o tempo entre incidentes com a mesma causa em todos os enlaces na rede da operadora. Enlaces sem incidentes ou com apenas um incidente de uma causa são contabilizados com tempo entre incidentes infinito. A figura mostra que, em geral, enlaces apresentam incidentes com a mesma causa com frequência muito baixa. (Note que o eixo y vai até 0,14.) A exceção são incidentes de rompimento, que é a causa mais comum (tabela 3) e reincidente.

5.5. Características de incidentes localização geográfica

A localização de um enlace, i.e., capital ou interior, é determinada de acordo com a localização da ponta mais distante da capital. Se um enlace interliga um concentrador em uma capital a um roteador de acesso de usuário no interior, consideramos que o enlace está no interior. De forma similar, se um enlace do *backbone* liga um PoP na capital a um PoP no interior, consideramos que o enlace está no interior. Utilizando esta classificação, 39% dos enlaces estão localizados no interior e totalizam 38% das falhas.

A figura 7(a) mostra que incidentes na capital e no interior têm tempos de solução similares. A figura 7(b) mostra que o tempo de recuperação em campo é menor para incidentes em capitais do que no interior. Este resultado é esperado, pois o tempo de recuperação em campo para enlaces inclui o tempo de deslocamento das equipes de campo. Como a rede do interior é mais espaçada geograficamente, o tempo de deslocamento de uma equipe técnica até o local do incidente é maior comparado ao tempo de deslocamento na capital. Consideramos a diferença de aproximadamente uma hora pequena, o que pode ser explicado pela boa distribuição geográfica das equipes de campo.

A figura 7(c) mostra que o tempo entre incidentes no interior é significativamente menor que nas capitais. (Note que o eixo y vai apenas até 0,4.) Este resultado pode ser



(a) Tempo de solução (h) (b) Tempo de trabalho em campo (h) (c) Tempo entre incidentes (dias)

Figura 7. Análise de incidentes por localização geográfica.

devido à diferença de infra-estrutura entre PoPs nas capitais e no interior. Apenas 39% dos enlaces estão no interior, espalhados em um número maior de PoPs. Outra explicação para a maior frequência de incidentes em enlaces no interior é que enlaces no interior ficam em produção por um período de tempo 31% menor que enlaces nas capitais (recorde que enlaces em produção por menor tempo apresentam maior frequência de falhas, figura 3(a)). Os equipamentos de acesso (como modems e switches) instalados pela operadora nos clientes são os mesmos, independentes da localização do cliente. Assim não consideramos que equipamentos são uma justificativa para as diferenças observadas.

6. Implicações práticas

Os resultados do presente trabalho podem ser aplicados para direcionar esforços na melhoria da operação de redes de telecomunicações, particularmente redes IP utilizando diferentes tecnologias de transmissão na camada de enlace.

Por exemplo, os resultados indicam que incidentes devidos a rompimento de fibra óptica possuem os maiores tempos de resolução. Esta informação pode motivar o desenvolvimento de novos procedimentos para resposta a este tipo de incidentes ou guiar o processo de expandão da rede para minimizar o impacto deste tipo de incidentes. De forma complementar, os resultados indicam que falhas em enlaces Metro Ethernet são as mais comuns. Este resultado motiva o desenvolvimento de ferramentas e processas que reduzam a frequência de incidentes nesses enlaces. Por último, os resultados podem ser utilizados para guiar o aprimoramento de sistemas de monitoramento, permitindo diagnóstico mais rápido e preciso de incidentes de rede.

7. Trabalhos relacionados

Caracterização de incidentes em redes. Outros trabalhos na literatura já caracterizaram incidentes em redes de trânsito. Apesar do objetivo comum de melhorar nosso entendimento sobre incidentes de rede, cada trabalho utiliza uma abordagem diferente, que depende dos dados disponíveis para estudo (que variam significativamente de uma rede para outra). Por exemplo, [Ghobadi and Mahajan 2016] utilizaram medições de qualidade de sinal para caracterizar falhas em enlaces ópticos enquanto [Govindan et al. 2016] utilizaram relatórios *post-mortem* de incidentes para entender suas causas e a importância de mecanismos que garantam disponibilidade da rede. Mais similar ao nosso trabalho são estudos de caracterização de registros de in-

cidentes [Markopoulou et al. 2008, Dainotti et al. 2011, Turner et al. 2010]. Alguns destes trabalhos utilizaram informações (*logs*) dos dispositivos da rede para corroborar as informações nos registros de incidentes; neste trabalho não tivemos acesso a *logs* dos dispositivos.

Monitoramento. Como informações sobre características de uma rede são essenciais para a operação da rede, execução de aplicações críticas e desenvolvimento de novas tecnologias, existe um grande esforço de pesquisa em ferramentas de monitoramento de rede. Os desafios estão em obter informações precisas a baixo custo (banda e processamaento) em um ambiente distribuído de larga escala, heterogêneo e repleto de fatores que interferem no monitoramento. Como exemplo, pesquisadores estudam soluções para detecção e localização de falhas [Katz-Bassett et al. 2012, Dhamdhere et al. 2007], congestionamento [Cardwell et al. 2016, Sommers et al. 2006], erros de configuração [Beckett et al. 2016, Gember-Jacobson et al. 2016], perda de pacotes [Sommers et al. 2007], *bufferbloat* [Gettys and Nichols 2011], dentre outros.

Ferramentas de monitoramento com e sem colaboração da rede. Ferramentas de monitoramento podem depender de colaboração por parte da rede monitorada, como acesso a informações coletadas por roteadores (e.g., [Mahimkar et al. 2009, Mahimkar et al. 2008, Shaikh and Greenberg 2004, Kompella et al. 2005]). Em geral, estas soluções obtém resultados mais precisos a menor custo, visto a melhor qualidade das informações. Estas ferramentas, porém, estão restritas aos operadores da própria rede. Os registros de incidentes caraterizados neste trabalho foram obtidos via uma parceria com a operadora de telecomunicações. Ferramentas de monitoramento que não dependem de colaboração por parte da rede monitorada coletam informações a partir de dispositivos externos (e.g., [Katz-Bassett et al. 2008, Kreibich et al. 2010, Chandrasekaran et al. 2015, Dischinger et al. 2010, Nikravesh et al. 2014, Quan et al. 2013]). Estas ferramentas trabalham com dados de menor visibilidade e são mais suscetíveis a interferências.

8. Conclusão

O crescimento da Internet e da demanda por serviços em rede dependem de avanço contínuo das tecnologias de comunicação e das soluções de monitoramento e gerência, essenciais para o rastreamento e solucionamento dos inevitáveis incidentes de rede. Neste trabalho caracterizamos um conjunto de registros de incidentes de uma na rede de uma operadora de telecomunicações de médio porte. Analisamos três características principais dos incidentes (tempo de solução, tempo de trabalho em campo e tempo entre incidentes) em quatro dimensões (tempo, tecnologia, causa e localidade geográfica).

Nossos resultados mostram que a frequência de incidentes em um enlace diminui à medida que permanece em produção. Nossos resultados mostram também que mudanças na frequência e no tempo de solução de incidentes são explicadas pela mudança no perfil das tecnologias utilizadas na rede: diferentes tecnologias apresentam frequência e tempo de solução de incidentes significativamente diferentes. Identificamos ainda possíveis pontos de melhora no processo de tratamento de incidentes, por exemplo, na identificação da causa de um incidente.

Nossos resultados apontam o impacto de cada dimensão nas características de incidentes, evidenciando problemas e possibilidades de melhoria. Nossos resultados podem ajudar operadores e pesquisadores a desenvolverem novas técnicas e processos para

tratamento de incidentes de rede.

Como trabalho futuro planejamos extender a nosso estudo para incluir *logs* coletados por comutadores e roteadores, complemetando os dados de registros e provendo melhor entendimento das causas e impactos de falhas.

Referências

Beckett, R., Mahajan, R., Millstein, T., Padhye, J., and Walker, D. (2016). Don'T Mind the Gap: Bridging Network-wide Objectives and Device-level Configurations. In *Proc. ACM SIGCOMM*.

Cardwell, N., Cheng, Y., Gunn, C. S., Yeganeh, S. H., and Jacobson, V. (2016). BBR: Congestion-Based Congestion Control. *ACM Queue*, 14(5).

Chandrasekaran, B., Smaragdakis, G., Berger, A., Luckie, M., and Ng, K.-C. (2015). A Server-to-server View of the Internet. In *Proc. ACM CoNEXT*.

Cisco (2011). The Economics of Networking. In *Technical Report*.

Cunha, I., Teixeira, R., Feamster, N., and Diot, C. (2009). Measurement Methods for Fast and Accurate Blackhole Identification with Binary Tomography. In *Proc. ACM IMC*.

Dainotti, A., Squarcella, C., Aben, E., Claffy, K. C., Chiesa, M., Russo, M., and Pescapé, A. (2011). Analysis of Country-wide Internet Outages Caused by Censorship. In *Proc. ACM IMC*.

Dhamdhere, A., Teixeira, R., Drovolis, C., and Diot, C. (2007). NetDiagnoser: Troubleshooting Network Unreachabilities Using End-to-end Probes and Routing Data. In *Proc. ACM CoNEXT*.

Dischinger, M., Marcon, M., Guha, S., Gummadi, K. P., Mahajan, R., and Saroiu, S. (2010). Glasnost: Enabling End Users to Detect Traffic Differentiation. In *Proc. USENIX NSDI*.

Gember-Jacobson, A., Viswanathan, R., Akella, A., and Mahajan, R. (2016). Fast Control Plane Analysis Using an Abstract Representation. In *Proc. ACM SIGCOMM*.

Gettys, J. and Nichols, K. (2011). Bufferbloat: Dark Buffers in the Internet. *Queue*, 9(11).

Ghobadi, M. and Mahajan, R. (2016). Optical Layer Failures in a Large Backbone. In *Proc. ACM IMC*.

Govindan, R., Minei, I., Kallahalla, M., Koley, B., and Vahdat, A. (2016). Evolve or Die: High-Availability Design Principles Drawn from Googles Network Infrastructure. In *Proc. ACM SIGCOMM*.

Katz-Bassett, E., Madhyastha, H., John, J. P., Krishnamurthy, A., Wetherall, D., and Anderson, T. (2008). Studying Black Holes in the Internet with Hubble. In *Proc. USENIX NSDI*.

Katz-Bassett, E., Scott, C., Choffnes, D. R., Cunha, I., Valancius, V., Feamster, N., Madhyastha, H. V., Anderson, T., and Krishnamurthy, A. (2012). LIFEGUARD: Practical Repair of Persistent Route Failures. In *Proc. ACM SIGCOMM*.

Kompella, R. R., Yates, J., Greenberg, A., and Snoeren, A. C. (2005). IP Fault Localization via Risk Modeling. In *Proc. USENIX NSDI*.

Kreibich, C., Weaver, N., Nechaev, B., and Paxson, V. (2010). Netalyzr: Illuminating The Edge Network. In *Proc. ACM IMC*.

Mahimkar, A., Yates, J., Zhang, Y., Shaikh, A., Wang, J., Ge, Z., and Ee, C. (2008). Troubleshooting Chronic Conditions in Large IP Networks. In *Proc. ACM CoNEXT*.

Mahimkar, A. A., Ge, Z., Shaikh, A., Wang, J., Yates, J., Zhang, Y., and Zhao, Q. (2009). Towards Automated Performance Diagnosis in a Large IPTV Network. In *Proc. ACM SIGCOMM*.

Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C. N., Ganjali, Y., and Diot, C. (2008). Characterization of Failures in an Operational IP Backbone Network. *IEEE/ACM Trans. Netw.*, 16(4):749–762.

Nguyen, H., Teixeira, R., Thiran, P., and Diot, C. (2009). Minimizing Probing Cost for Detecting Interface Failures: Algorithms and Scalability Analysis. In *Proc. IEEE INFOCOM*.

Nikravesh, A., Choffnes, D. R., Katz-Bassett, E., Mao, Z. M., and Welsh, M. (2014). Mobile Network Performance from User Devices: A Longitudinal, Multidimensional Analysis. In *Passive and Active Measurement Conference*.

Peter, S., Javed, U., Zhang, Q., Woos, D., Krishnamurthy, A., and Anderson, T. (2014). One Tunnel is (Often) Enough. In *Proc. ACM SIGCOMM*.

Quan, L., Heidemann, J., and Pradkin, Y. (2013). Trinocular: Understanding Internet Reliability Through Adaptive Probing. In *Proc. ACM SIGCOMM*.

Shaikh, A. and Greenberg, A. (2004). OSPF Monitoring: Architecture, Design and Deployment Experience. In *Proc. USENIX NSDI*.

Sommers, J., Barford, P., Duffield, N., and Ron, A. (2007). Accurate and Efficient SLA Compliance Monitoring. In *Proc. ACM SIGCOMM*.

Sommers, J., Barford, P., and Willinger, W. (2006). A Proposed Framework for Calibration of Available Bandwidth Estimation Tools. In *Proc. IEEE Symp. on Comp. and Comm.*

Turner, D., Levchenko, K., Snoeren, A., and Savage, S. (2010). California Fault Lines: Understanding the Causes and Impact of Network Failures. In *Proc. ACM SIGCOMM*.

Trilha Principal do SBRC 2017
Sessão Técnica 19
Virtualização de Funções de Rede

Emprego de NFV e Aprendizagem por Reforço para Detectar e Mitigar Anomalias em Redes Definidas por Software

Pedro H. A. Faustini[1], **Anderson S. Silva**[1], **Lisandro Z. Granville**[1], **Alberto E. Schaeffer-Filho**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{phafaustini,assilva,granville,alberto}@inf.ufrgs.br

Abstract. *A computer network is subject to several anomalies, and it is necessary to coordinate detection and mitigation techniques to keep the network operational. This paper proposes the use of reinforcement learning to promote resilience in software-defined networking (SDN). In particular, it is proposed collecting network metrics and their grouping into profiles, each one having a set of actions to handle problems using reinforcement learning, network functions virtualisation (NFV), and an SDN controller. Policies for dealing with anomalies are defined, based on the rewards for each action. Results show that the system obtains mostly positive rewards, but a small increment in the topology size leads to more than four times the number of entries in the state-action table.*

Resumo. *Uma rede de computadores está sujeita a diversas anomalias, e é necessário coordenar técnicas de detecção e mitigação para mantê-la operacional. Este artigo propõe aprendizagem por reforço para promover resiliência em redes definidas por software (SDN). Propõe-se que métricas de rede sejam coletadas e agrupadas em perfis, cada um com um conjunto de ações que trate problemas usando aprendizagem por reforço, virtualização de funções de rede (NFV) e um controlador SDN. Políticas para lidar com anomalias são definidas com base nas recompensas das ações. Os resultados mostram que o sistema obtém majoritariamente recompensas positivas, mas um pequeno aumento na topologia mais do que quadriplica o número de entradas na tabela estado-ação.*

1. Introdução

Redes de computadores podem apresentar diversas anomalias. Entende-se por anomalias não somente a presença de fluxos com conteúdo malicioso, como no caso de ataques de negação de serviço, mas também fluxos benignos que, combinados, podem deteriorar a experiência dos usuários. Surge portanto a necessidade de manter a resiliência da rede, isto é, a habilidade da rede em manter um nível aceitável de operação frente a várias falhas e desafios à sua operação normal [Sterbenz et al. 2010]. Diferentes anomalias requerem diferentes técnicas de mitigação, o que leva ao problema de como coordenar essas técnicas harmonicamente, ou seja, como acionar uma técnica sem que sua ação prejudique ações de outras técnicas usadas na promoção da resiliência.

Este artigo tem como objetivo investigar como redes definidas por software (*Software-Defined Networking* - SDN) podem se beneficiar de técnicas baseadas em

aprendizagem de máquina e virtualização de funções de rede (*Network Functions Virtualisation* - NFV) de modo a selecionar estratégias de mitigação para diferentes tipos de anomalias. Por exemplo, a sobrecarga de um servidor devido ao alto número, ainda que legítimo, de requisições, ou o gargalo em um enlace que transmite muitos pacotes de diferentes máquinas, são considerados anomalias e devem ser tratados.

Nesse contexto, as principais contribuições deste artigo são: **(i)** levantamento do estado-da-arte sobre o uso de aprendizagem por reforço em redes; **(ii)** modelagem de um agente que adota técnicas de aprendizagem por reforço e aprende a lidar com diferentes tipos de anomalias e **(iii)** implementação de um protótipo e avaliação de aspectos como crescimento das tabelas e desempenho na obtenção das recompensas.

Acreditamos que a combinação dos paradigmas de SDN e NFV tem o potencial de facilitar a construção de técnicas mais flexíveis para a promoção de resiliência. Em específico, SDN fornece um ambiente propício para manipular regras de roteamento de pacotes, além de facilitar a coleta de métricas em função da presença do controlador, que possui visão global da rede [McKeown et al. 2008]. NFV, por sua vez, fornece flexibilidade na instanciação de mecanismos a serem usados na detecção e mitigação de ameaças.

O restante deste artigo segue a seguinte estrutura: a Seção 2 apresenta o referencial teórico para os conceitos de SDN, NFV e aprendizagem por reforço. Na Seção 3 é proposto o modelo de trabalho conjunto de NFV e aprendizagem por reforço para promover a resiliência em SDN. Na Seção 4, são apresentados o protótipo implementado e resultados. A Seção 5 contém os trabalhos relacionados, e a Seção 6 apresenta as conclusões e prospecções de trabalhos futuros.

2. Fundamentação

Nesta seção serão apresentados importantes conceitos que formam a fundamentação deste trabalho. Primeiro será visto SDN, seguido de NFV e, por fim, aprendizagem por reforço.

2.1. Redes Definidas por Software

O conceito de SDN fornece um novo paradigma para o gerenciamento de recursos de rede. Uma das suas mais destacadas características é a figura de um software controlador que centraliza a lógica de funcionamento da rede, desacoplando o plano de controle do plano de dados [ONF 2014]. O plano de dados é responsável por trafegar pacotes, e é mais dependente de componentes de hardware, como switches. O plano de controle possui a capacidade de executar funções via software a fim de flexibilizar a inserção e remoção de funcionalidades de controle. Um elemento fundamental no plano de controle é o controlador, que possui visão global da rede e assim é capaz de otimizar o gerenciamento de fluxo com flexibilidade e escalabilidade. Um exemplo é a possibilidade de se alterar dinamicamente estratégias de roteamento.

O protocolo OpenFlow é o padrão *de facto* para comunicação entre plano de controle e plano de dados [B. A. A. Nunes et al. 2014]. O protocolo serve como um canal entre o plano de dados e o plano de controle, tornando possível adicionar ou remover entradas nas tabelas de fluxos dos switches. Além disso, redes programáveis podem diminuir barreiras para a concepção de novas ideias, facilitando a inovação em redes [McKeown et al. 2008].

2.2. Virtualização de Funções de Rede

Redes de computadores costumam possuir muitas funções em sua infraestrutura. Exemplos de funções de rede incluem *firewalls*, sistema de detecção de intrusões, monitores, *honeypots* e outros. O problema é que tais equipamentos, geralmente construídos em hardware dedicado, rapidamente atingem o fim da vida útil. Além disso, mesmo enquanto apresentam desempenho satisfatório, podem facilmente ser ultrapassados por outras tecnologias mais recentes [Herrera e Botero 2016]. Outro problema é a falta de flexibilidade no gerenciamento desses dispositivos. NFV busca portar tais funções para hardware de propósito geral, executando-as em ambientes virtualizados, como máquinas virtuais. Entre os benefícios estão redução de custos, escalabilidade e flexibilidade para inovação. Contudo, a troca de hardware dedicado para máquinas de propósito geral traz desafios. Entre os principais estão o desempenho em ambientes virtualizados, resiliência frente a falhas tanto de hardware como de software, entre outros [ETSI 2012].

Há três componentes principais na arquitetura NFV: infraestrutura, serviço e orquestração & gerenciamento [Herrera e Botero 2016]. A Infraestrutura, conhecida como NFVI (*NFV Infrastructure*), abrange os recursos de hardware e software, bem como o ambiente de virtualização. O componente de Serviço abrange as funções de rede virtualizadas, ou VNFs (*Virtualised Network Functions*). Elas são executadas em ambientes virtualizados, em vez de hardware dedicado (*middleboxes*) [King et al. 2015]. Orquestração e gerenciamento, conhecida como NFV-Mano (*management and orchestration*), foca em tarefas mais específicas da virtualização. Por exemplo, se entre as VNFs estão *firewall* e um detector de intrusões (IDS), NFV-Mano pode determinar que um pacote siga, após o resultado da inspeção do detector, para um *firewall*. Apesar dos conceitos de NFV e SDN serem independentes, as tecnologias podem ser complementares. É possível que, no futuro, ambas tecnologias se tornem menos distinguíveis até se unirem como um único paradigma em redes baseadas em software [ETSI 2014].

2.3. Aprendizagem por Reforço

Aprendizagem por reforço envolve um agente que analisa um ambiente e interage com ele realizando ações. A partir de uma ação tomada no instante *t*, no próximo momento *t+1* o agente muda de estado e recebe uma recompensa [Sutton e Barto 2012]. Uma maneira de definir problemas de aprendizagem por reforço é através dos processos de decisão de Markov (*Markov decision process*, ou MDP). MDP fornece um *framework* matemático muito usado na modelagem das dinâmicas de um ambiente sob diferentes ações, e pode ser definido por uma 4-upla [Malialis 2014] $\langle S, A, R, P \rangle$, em que ***S*** é o conjunto de estados, ***A*** é o conjunto de ações, ***R*** é função de recompensa (retorna a recompensa no estado s_{t+1}, quando a ação a é executada no estado s), e ***P*** é a função de probabilidade de transição (retorna a probabilidade de se atingir o estado s_{t+1} quando a ação a é executada no estado s). As funções de recompensa e probabilidade compõem a dinâmica do ambiente.

Na maioria dos problemas, a dinâmica do ambiente não está disponível [Malialis 2014], e os problemas de aprendizagem por reforço costumam envolver a estimação de funções valor. Uma tabela estado-ação, ou Q-Table $Q(s, a)$, procura definir o quão bom é executar uma ação a em um estado s. Esta tabela é atualizada a partir da recompensa obtida pelo agente. Entre as formas de atualização da recompensa está o algoritmo Sarsa [Sutton e Barto 2012]. Cada letra de seu nome é uma alusão aos elementos que compõem a aprendizagem: $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$, ou estado e ação no *time*

step atual, a recompensa adquirida no *time step seguinte*, e o par estado-ação no estado seguinte. O algoritmo, em pseudo-código, é apresentado a seguir:

Algoritmo 1: SARSA

Input: Q-table
Output: Q-table atualizada
$\forall s \in S, \forall a \in A, Q(s,a) = 0$ /* Inicializar Q(s,a) */
foreach *episódio* **do**
    Escolha $a \in A$ disponível em $s \in S$ segundo uma política (e.g. ϵ-*greedy*)
    foreach *time step do episódio* **do**
        Execute a ação a, observe s', r
        Escolha $a' \in A$ disponível em $s' \in S$ segundo uma política (e.g. ϵ-*greedy*)
        $Q(s,a) \leftarrow Q(s,a) + \alpha[r - Q(s,a)]$
        $S \leftarrow S', A \leftarrow A'$

A recompensa é denotada por *r* e α é a taxa de aprendizagem. Quanto maior o seu valor, maior será a variação dos valores atualizados na tabela a partir da recompensa. Enquanto aprende, o agente se depara com um dilema entre adotar uma ação cuja recompensa é conhecida ou explorar novas possibilidades e adquirir mais conhecimento sobre as consequências de suas ações. Uma técnica popular e efetiva que lida com este dilema é ϵ-*greedy* [Sutton e Barto 2012]. Nela, um fator ϵ indica o quanto um agente irá preferir testar novas possibilidades frente a opções já conhecidas. Por exemplo, se $\epsilon = 0.1$, significa que em 10% dos casos o agente tentará uma abordagem desconhecida.

3. Aprendizagem por Reforço em Infraestruturas Baseadas em SDN/NFV

Neste artigo, propomos um modelo de aprendizagem por reforço integrado à arquitetura de NFV por meio de um agente que reside no orquestrador. O agente recebe métricas de rede e constrói perfis de rede. Cada perfil busca tratar uma anomalia. Cada perfil é formado por um conjunto de ações, bem como uma tabela de estados e uma tabela estado-ação. A cada ciclo de aprendizagem (um intervalo de tempo, ou *time step*, arbitrário), o agente seleciona um perfil, executa uma ação disponível e recebe uma recompensa para atualizar a tabela estado-ação (Q-Table) daquele perfil, conforme exibe a Figura 1.

A cada ciclo, o agente executa somente uma ação de um perfil. Isso acontece para evitar que, ao executar duas ou mais ações em um mesmo ciclo, uma delas afete os efeitos da outra. A cada perfil pode ser conferida uma prioridade, de forma que se uma anomalia for detectada em dois ou mais perfis, o agente dará prioridade a um deles ao escolher uma ação para mitigá-la.

Dois perfis podem ser mesclados para o sistema funcionar harmonicamente. Por exemplo, um determinado perfil de rede pode ter, em seu conjunto de ações, a busca por caminhos alternativos entre dois hosts caso a rota adotada apresente alto tráfego por causa da presença de outros fluxos. Outro perfil pode ter entre suas ações a instanciação de uma réplica de um serviço ou função de rede em outro ponto da topologia (p.ex. em uma CDN). Entretanto, replicar um conteúdo implica em desviar a rota de certos hosts

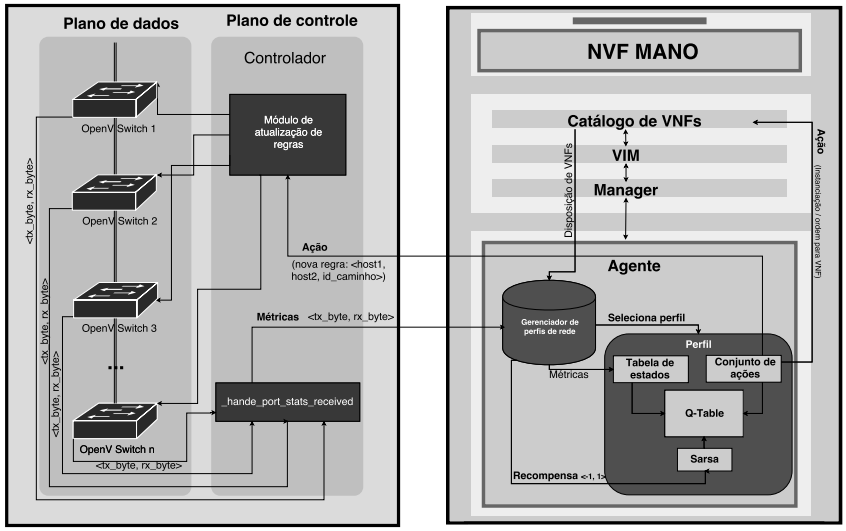


Figura 1. Arquitetura para aprendizagem por reforço em infraestruturas baseadas em SDN/NFV

para ele. Nestes casos, os perfis devem ser unidos, de forma que tenham a mesma prioridade e o agente aprenda quais ações são melhores para quais situações (neste exemplo, somente desviar a rota, ou instanciar um novo recurso e desviar a rota), por meio de uma recompensa em comum. Em específico, neste trabalho são estudados três perfis: **(i)** balanceamento de tráfego, **(ii)** replicação de servidores e **(iii)** ataques de negação de serviço.

3.1. Balanceamento de Tráfego

Dispositivos de encaminhamento, ou switches, possuem portas por onde links responsáveis por transmitir pacotes se conectam. Se os links incidentes a um switch apresentam alta vazão, a fila de espera (*buffer*) do switch pode encher, ocasionando o descarte de pacotes e prejudicando a experiência de usuários. Portanto, a fim de evitar essa sobrecarga, pode ser interessante encaminhar pacotes por caminhos alternativos. A ideia é que, caso esteja menos congestionada, uma rota mais longa pode ser mais vantajosa do que outra mais curta. As ações deste perfil consistem em **(i)** ação vazia, **(ii)** aplicar o caminho mínimo entre dois hosts, e **(iii)** alterar um caminho aplicado entre dois hosts.

Um caminho é dito congestionado (tem status alto) se algum de seus links apresenta tráfego igual ou superior a um determinado percentual da capacidade máxima, por exemplo 80%. Senão, o status é considerado médio se algum de seus links apresenta tráfego em uma faixa inferior, por exemplo, a partir de 30% da capacidade máxima. Senão, é considerado baixo. Um link pode pertencer a vários caminhos. Se o tráfego gerado por cada fluxo for pequeno, mas somados deixarem o link sobrecarregado, ambos caminhos apresentarão status alto.

O agente, a cada ciclo de aprendizagem, coleta informações das tabelas de fluxo para determinar quantos bytes chegaram aos switches por meio de cada link. Para fazer isso, o agente soma os campos *tx_byte*[1] e *rx_byte*[2] de todas entradas de mesma *in_port*.

[1]Bytes transmitidos por aquela porta
[2]Bytes recebidos por aquela porta

Os campos *tx_byte* e *rx_byte* das tabelas de fluxo são cumulativos, isto é, não possuem a informação de quantos bytes trafegaram por uma porta em um determinado tempo. Contudo, tal restrição é facilmente contornada calculando a diferença entre os valores encontrados nos ciclos atual e anterior. Em cada *time step*, o agente tem a sua disposição o estado atual e o conjunto de caminhos possíveis calculado pelo controlador para cada par de hosts. De posse dessas informações, ele pode solicitar que o controlador modifique regras nas tabelas de fluxo dos switches de forma a trocar um caminho por outro.

O agente somente poderá alterar um caminho aplicado entre dois hosts caso um ou mais links em um caminho esteja no status alto. Existem dois objetivos: *(i)* evitar que um link esteja com status alto da sua capacidade total de vazão, e *(ii)* adotar sempre que possível o menor caminho possível. A recompensa conferida ao agente é de 1 se após executar a ação não houver caminhos que apresentem links com status alto, ou -1 caso contrário. Ou, formalmente:

$$R = \begin{cases} -1, & \text{se há link com tráfico em status alto} \\ 1, & \text{caso contrário} \end{cases} \tag{1}$$

Intuitivamente, o agente procurará realizar o balanceamento de forma que não haja sobrecarga nos links. Para isso, pode ser necessário alterar a rota de determinados fluxos de forma que eles não sigam o caminho mínimo entre origem e destino. Entretanto, em algum momento o agente retornará os fluxos desviados para o caminho mínimo. Ele pode fazer isso prematuramente, e ser punido por isso, ou recompensado. Com o passar dos ciclos, aprenderá quais estados permitem o retorno ao caminho mínimo.

3.2. Replicação de Servidores

Servidores fornecem serviços para clientes. Por mais que um determinado serviço fornecido pelo servidor possa ser paralelizado para atender diversos pedidos, o servidor pode ficar sobrecarregado caso receba muitas requisições. Para contornar este problema, VNFs que replicam este serviço são instanciadas em outros pontos na rede e absorvem parte das requisições que sobrecarregavam o link de acesso ao servidor. A construção do perfil de carga sobre servidores ocorre por meio de três métricas: **(i)** o tráfego no link de acesso entre servidor e switch, **(ii)** a relação de servidores ou VNFs de réplicas, e **(iii)** os caminhos incidentes ao servidor ou réplicas.

Como o tráfego em um link possui muitos valores possíveis, ele é dividido em três faixas: *baixo* (por exemplo, $[0\%, 30\%)$ de uso), *médio* (por exemplo, $[30\%, 80\%)$) e *alto* (por exemplo, $[80\%, 100]$). A relação de VNFs é coletada pelo próprio catálogo de VNFs do orquestrador NFV. As VNFs não são, contudo, colocadas de forma totalmente aleatória pela rede. Elas são conectadas somente aos switches que fazem parte de caminhos utilizados para se chegar ao servidor. O número de estados possíveis dependerá do tamanho da rede: quanto maior ela for, mais estados podem existir da combinação das três métricas supracitadas. A tabela ganha entradas a medida em que o número de VNFs é alterada, a carga no link de acesso muda, ou caminhos são modificados. Naturalmente, não há a necessidade de pré-calcular todas as combinações possíveis, e sim utilizar estruturas de dados que dinamicamente acrescentem os novos estados à medida em que eles forem verificados pela primeira vez, como listas. Para o perfil de carga nos servidores,

as ações possíveis consistem em: **(i)** instanciar uma réplica de servidor, **(ii)** remover uma réplica instanciada, e **(iii)** ação vazia.

A remoção de uma réplica só ocorre quando o link de acesso ao servidor reporta baixo tráfego. Quando há alta vazão, espera-se uma necessidade maior de instanciações do que em uma situação de média vazão, por exemplo. O agente vai aprender quantas réplicas são necessárias instalar e em quais switches para balancear o tráfego da rede de acordo com a carga sobre o link de acesso dos servidores. O agente tem dois objetivos: *(i)* manter o link de acesso do servidor ao switch em um estado de baixo tráfego, *(ii)* fazer isso usando o mínimo de recursos possível. A recompensa conferida ao agente é de 1 se após executar a ação o link de acesso do servidor ao switch estiver em um estado de baixo tráfego ou -1 caso contrário. Ou, formalmente:

$$R = \begin{cases} 1, & \text{se tráfego em link de acesso for baixo} \\ -1, & \text{caso contrário} \end{cases} \tag{2}$$

Intuitivamente, conforme o volume de tráfego varia, o agente procura a configuração mínima de VNFs que alivie o estresse sobre os links de acesso dos servidores de forma a mantê-los com status de baixa vazão. É possível que o agente execute uma ação que leve um link que apresentava baixo consumo para médio consumo, por exemplo. Isso acontece porque tal link estava com baixo nível de tráfego por causa das VNFs instanciadas. Ao explorar uma configuração com menos VNFs para aquela situação de tráfego, o link ficou sobrecarregado e o agente foi punido com uma recompensa negativa. No longo prazo, aprenderá a como reagir frente a cada configuração de tráfego.

3.3. Mitigação de Ataques

Ao contrário de balanceamento de carga ou replicação de servidores, em que o agente deve aprender a melhor maneira de balancear o tráfego ou onde instanciar uma réplica, em um cenário de ataque não existe outra alternativa senão mitigar os fluxos maliciosos. Tentar absorver o ataque balanceando o tráfego pode se tornar uma tarefa impraticável, ou demandar a instanciação de muitos recursos. Faz sentido que este perfil receba uma prioridade maior do que os outros dois abordados anteriormente. O agente deve ser capaz de distinguir fluxos maliciosos de legítimos, de forma que mitigue os ataques e execute ações dos demais perfis somente para os pacotes legítimos. Este perfil não adota aprendizagem de máquina em sua solução, e assim não há uma tabela estado-ação. Este perfil possui somente dois estados, portanto: **(i)** malicioso e **(ii)** benigno.

Quando este perfil se encontra no estado malicioso, a única ação possível é mitigar o ataque. Enquanto um ataque não for detectado, o agente pode executar ações de perfis de prioridade mais baixa. O controlador mantém registro de média e desvio padrão de requisições que cada servidor recebe. Ele obtém essa informação verificando nas tabelas de fluxo dos switches quais fluxos são destinados a quais servidores. O controlador também mantém em uma tabela um registro a respeito de a quais switches cada host, identificado pelo endereço IP, se conecta. Uma aplicação executando no plano de aplicações de SDN coleta essas informações e, caso em três *time steps* consecutivos o número de requisições recebido por um servidor seja superior a sua média mais um desvio, um ataque é considerado em curso. Técnicas mais avançadas de detecção e mitigação de ataques

de negação de serviço existem, mas para o escopo deste trabalho se adota uma heurística mais simples para lidar com eles. O próximo passo da aplicação é mitigar o ataque.

Para a mitigação, foi adotada uma heurística simples, para fins de prova de conceito: sendo r o número normal de requisições que o servidor recebe, e c o número de clientes que está conectado a ele, a aplicação assume que em média cada cliente deve solicitar r/c requisições. Clientes acima desse limiar mais três desvios são barrados. Para isso, busca-se no fluxo a informação relativa ao endereço IP de origem desses clientes e consulta-se a tabela para saber em quais switches eles estão conectados. O agente então instancia e ordena que VNFs que implementam *firewalls* barrem pacotes provenientes dos endereços IPs considerados suspeitos.

4. Protótipo e Resultados

Esta seção descreve o protótipo implementado, o qual está disponível publicamente[3]. Em específico, na Subseção 4.1 são vistos detalhes do protótipo desenvolvido, e na Subseção 4.2 são apresentados experimentos realizados e resultados obtidos.

4.1. Implementação do Protótipo

O protótipo desenvolvido abrange os perfis de mitigação de ataques e balanceamento de tráfego. Para a realização de experimentos sobre o sistema proposto, foi utilizada a plataforma Mininet[4], que emula uma SDN, e o framework Pox[5], que implementa a versão 1.0 do OpenFlow, para o desenvolvimento do controlador. Contudo, Mininet não provê suporte nativo para NFV. As funções de rede portanto foram implementadas em contêineres Docker[6]. Seus contêineres são baseados em Linux, compartilham o kernel do sistema hospedeiro, mas proveem isolamento entre as aplicações como no caso de máquinas virtuais.

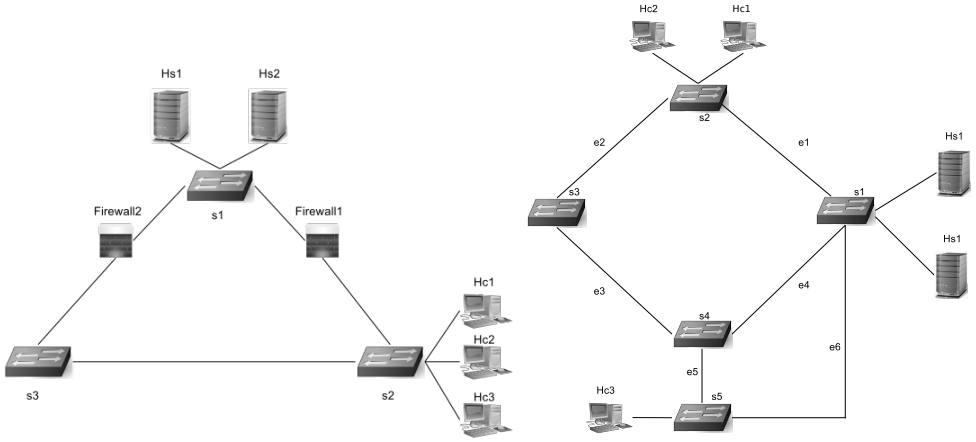
Docker provê a infraestrutura básica para hospedar funções de rede, mas não a orquestração das mesmas. Na implementação feita, a partir do projeto Containernet[7], que fornece ao Mininet suporte a contêineres Docker, esta tarefa é desempenhada conjuntamente por controlador e agente. O controlador é quem determina quais pacotes passem por *firewalls* instanciados, por exemplo, antes de chegar ao seu destino. Também é ele quem coleta métricas e as envia para o agente construir os perfis e estados da rede. É o agente, por outro lado, quem realiza as chamadas necessárias para instanciar as VNFs definidas por *Dockerfiles* e ordena que um *firewall* barre determinados fluxos. A instanciação é feita por meio de uma chamada especial, *net.addDocker*. O agente também envia ao controlador solicitação de ações como a alteração da rota entre um par de hosts. O controlador monitora as tabelas de fluxos dos switches e envia as métricas necessárias para a formação dos estados em cada *time step* ao agente. O controlador também calcula os caminhos possíveis entre os hosts e envia essa relação ao agente.

4.2. Análise Experimental

Foram realizados dois experimentos. Em ambos, servidores estão em contêineres Docker e enviam arquivos de vídeo a clientes para gerar tráfego através de conexão TCP. Clientes

[3]https://github.com/phfaustini/containernet
[4]http://mininet.org/
[5]http://sdnhub.org/tutorials/pox/
[6]https://www.docker.com/what-docker
[7]https://github.com/containernet/containernet

são hosts usuais do Mininet. Links que conectam clientes e servidores a switches têm vazão máxima de 5 Mbps. Os demais trafegam até 11 Mbps. A banda consumida pelos clientes é 5 Mbps, que é o limite para os links que conectam hosts a switches. Para realizar os experimentos, foi utilizado um computador com processador Intel i7 de 3,4 Ghz e 16 GB de Ram, em uma máquina virtual executando o sistema operacional Ubuntu 14.04 (dos quais 10 GB Ram foram alocados para ela). Também em ambos os casos, foi adotada uma taxa de aprendizagem $\alpha = 0.1$ (para evitar que uma ação isolada possa impactar fortemente a política de decisões do agente), taxa de exploração iniciando em $\epsilon = 0.4$ (pois espera-se explorar prioritariamente uma ação conhecida, mas não raramente descobrir as consequências de outras ações) e dois segundos para cada *time step*.



(a) Topologia usada no experimento #1. (b) Topologia usada no experimento #2

Figura 2. Topologias usadas nos experimentos

No primeiro experimento, há dois servidores (Hs_1 e Hs_2) e três clientes (Hc_1, Hc_2 e Hc_3), conforme mostra a Figura 2 (a). O agente procura balancear o tráfego em direção aos servidores e precisa também mitigar um ataque. Neste caso, o perfil de mitigação possui prioridade mais alta em relação ao perfil de balanceamento. Duas VNFs que implementam um *firewall* têm a função de proteger os servidores de um eventual ataque vindo através dos switches *s2* e *s3*. Elas foram implementadas em contêineres por meio das ferramentas *bridge-utils*[8] e *ebtables*[9]. Também foi estipulado que após 40 *time steps* a taxa de exploração cairia para $\epsilon = 0.1$.

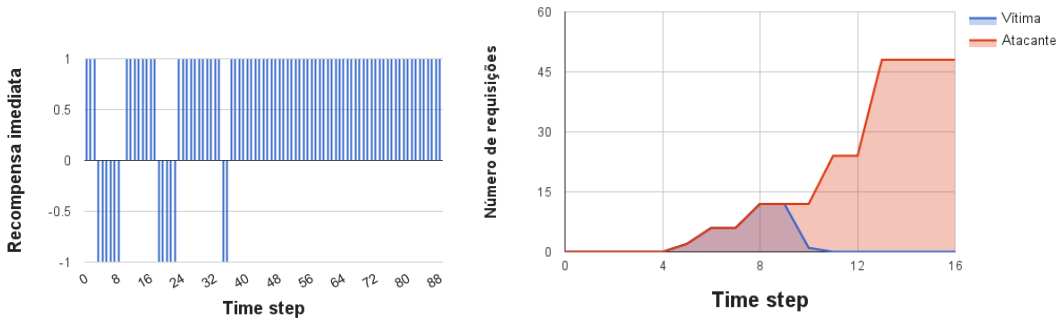
Os clientes Hc_1 e Hc_3 estabelecem conexão com o servidor Hs_1, e o cliente Hc_2 faz o mesmo com o servidor Hs_2. Inicialmente a troca de pacotes ocorre por meio do link que conecta os switches *s1* e *s2*. O arquivo trocado em cada conexão tem 150 Mb. O cliente Hc_3 é malicioso e executa diversas requisições TCP a fim de exaurir o servidor Hs_1. Rapidamente os links ficam sobrecarregados em função dos muitos arquivos sendo transmitidos, e o agente não encontra uma rota alternativa que ocasione uma recompensa positiva. O ataque é posteriormente detectado, e o agente ordena que as VNFs de *firewall* barrem todo o fluxo envolvendo o cliente Hc_3.

A Figura 3 (a) mostra as recompensas imediatas obtidas pelo agente para o perfil

[8]https://wiki.linuxfoundation.org/networking/bridge
[9]http://ebtables.netfilter.org

de balanceamento de carga. Durante a identificação e eliminação do ataque, no *time step* 9, não houve recompensa porque o perfil de rede ativo era o de mitigação, que possui prioridade mais alta em relação ao de balanceamento. Depois que a taxa de aprendizagem foi reduzida para 0.1, no *time step* 40, conforme o estipulado, observa-se que o agente somente adquiriu recompensas positivas. Isso evidencia que de fato encontrou ações que evitem o congestionamento dos links, caso contrário receberia recompensas negativas. Sobre os ataques, a Figura 3 (b) mostra a quantidade de requisições feitas pelo cliente malicioso Hc_3 (em vermelho) e o número de requisições que de fato chegaram ao servidor Hs_1 (em azul). Até o *time step* 9 havia paridade entre requisições do cliente e atendimentos por parte do servidor. Contudo, nota-se a atuação dos *firewalls*, que neste instante bloquearam o fluxo do atacante. Assim, a vítima não mais recebeu as requisições, apesar das tentativas persistentes do atacante.

O segundo experimento aborda somente balanceamento de tráfego, mas em uma topologia maior, permitindo mais combinações de rotas alternativas, conforme mostra a Figura 2 (b). Os hosts Hs_1 e Hs_2 estão conectados ao switch *s1* e agem como servidores, ao passo que Hc_1 e Hc_2 estão conectados no switch *s2* e são configurados como clientes.



(a) Recompensas imediatas obtidas. (b) Requisições enviadas e recebidas.

Figura 3. Recompensas e requisições

Um arquivo de 500 Mb é transmitido entre os hosts Hs_1 e Hc_1 e outro idêntico entre os hosts Hs_2 e Hc_2 ao mesmo tempo. O caminho inicial adotado pelo controlador para o encaminhamento dos pacotes é o caminho mínimo. Após 100 *time steps*, o terceiro cliente fazia o mesmo com o servidor Hs_1. A Figura 4 mostra as recompensas imediatas obtidas ao longo do tempo pelo agente, durante o balanceamento de carga.

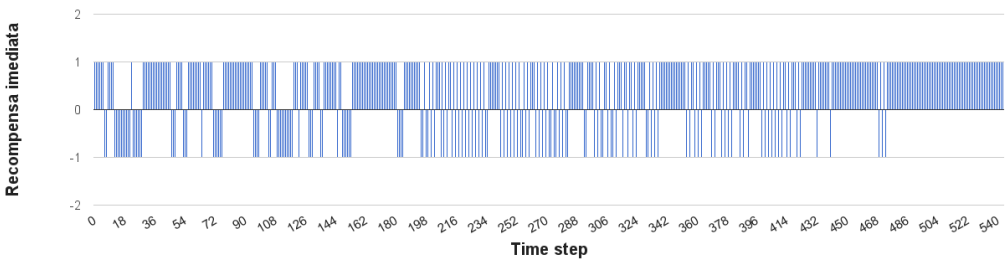
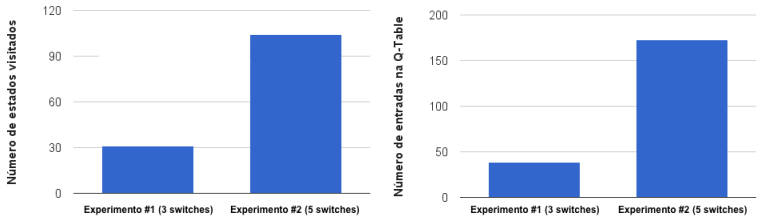


Figura 4. Recompensas imediatas obtidas ao longo do tempo

Após 400 *time steps*, a taxa de exploração foi reduzida para $\epsilon = 0.1$, e o agente

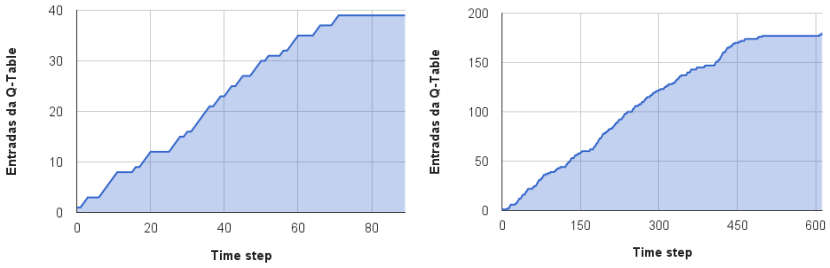
conseguiu encontrar um caminho que evite gargalos, obtendo recompensas positivas, conforme mostra a Figura 4. Em 10% dos casos, ele explorava ações desconhecidas, o que o levava a recompensas negativas, mas em proporção muito menor em relação ao início do experimento.

Analisando-se os dois experimentos, registra-se o baixo desempenho espacial da técnica de se armazenar estados em tabelas. O primeiro experimento apresentava uma topologia com três switches, e o segundo acrescentava dois, totalizando cinco switches. Apesar do pequeno aumento da topologia, o número de combinações possível para encaminhar pacotes cresceu de tal forma que o agente percorreu muito mais estados, conforme mostra a Figura 5 (a). Isso se refletiu ainda no tamanho da tabela estado-ação, que também registrou aumento expressivo no número de entradas adicionadas, conforme mostra a Figura 5 (b).



(a) Número de estados visitados (b) Tamanho da tabela estado-ação

Figura 5. Tamanho da tabela estado-ação e número de estados visitados



(a) Q-Table no experimento #1 (b) Q-Table no experimento #2

Figura 6. Crescimento da tabela estado-ação

Quanto mais métricas forem coletadas para a formação de um estado, com mais intensidade esse fenômeno tende a ocorrer. No caso do balanceamento de tráfego, por exemplo, ainda que os valores que compõem os estados sejam discretizados em faixas do tipo “alto”, “médio” e “baixo”, cada estado é representado por uma matriz, sendo que cada linha da matriz possui dois campos (caminho e status do caminho). Quanto maior o número de switches, mais caminhos possíveis entre os hosts existirão, levando a uma grande combinação de estados possíveis. O armazenamento dessas informações e o tempo necessário para o treinamento do agente pode ser custoso em uma rede de maior porte.

As figuras 6 (a) e 6 (b) mostram o quanto o número de entradas na Q-Table aumenta conforme o progresso dos *time steps* para os dois experimentos. Nota-se uma curva de crescimento acelerado, o que evidencia a necessidade de se pensar alternativas para o

armazenamento de dados em tabelas a fim de evitar, ou atenuar, esse fenômeno. Como a taxa de exploração foi reduzida ao final dos dois experimentos, o agente parou de explorar ações desconhecidas. Consequentemente, novas combinações de tráfego/ação pararam de ser acrescentadas à Q-Table, e houve um estancamento no seu crescimento.

5. Trabalhos Relacionados

Diversos pesquisadores têm se empenhado em tornar redes de computadores mais seguras. Técnicas de inteligência artificial têm sido adotadas para tornar redes mais resilientes, sejam elas SDN ou não. O mesmo vale para trabalhos que mesclam características de NFV e SDN. Esta seção seleciona publicações que abordam ataques de negação de serviço, que adotam aprendizagem por reforço para realizar balanceamento de tráfego e que procuram identificar anomalias em redes baseadas em infraestruturas SDN/NFV.

Machado, Granville e Schaeffer-Filho (2016) propõem uma arquitetura que combina SDN e NFV para promover resiliência. VNFs e controlador monitoram e mitigam ataques com base em um *control-loop* que identifica a melhor estratégia para resolver um tipo específico de anomalia na rede. O *control-loop*, que exerce a função de "cérebro" do sistema, não adota aprendizagem de máquina. Esta possibilidade é, contudo, citada para um trabalho futuro a fim de melhorar o *feedback* do *control-loop*.

Belyaev e Gaivoronski (2014) utilizam balanceamento de carga para lidar com ataques DDoS. Em sua abordagem, fluxos maliciosos não são descartados; em vez disso são redirecionados na rede. Seus experimentos mostraram que a solução ajuda a aumentar o tempo de sobrevivência a um ataque, ao tentar absorvê-lo, aproveitando a visão global que o controlador SDN tem para realizar a distribuição do tráfego. Nosso trabalho, por outro lado, optou por uma abordagem diferente, de separar a rede em perfis, e só balancear o tráfego de fluxos legítimos, eliminando aqueles considerados maliciosos primeiro com o auxílio da tecnologia NFV.

Malialis (2014) propõe uma abordagem para lidar com ataques de negação de serviço. Seu trabalho emprega aprendizagem por reforço instalando múltiplos agentes em roteadores. Sua arquitetura é descentralizada para ser mais resiliente a ataques e os agentes trabalham em conjunto descartando pacotes para manter um servidor operacional. Os agentes são dispostos em pontos fixos na topologia seguindo a proposta de Yau (2005). Sua tese não reside no universo de SDN nem NFV, mas não seria infactível, em um trabalho futuro, instalar os agentes em VNFs, em vez de roteadores. Isso conferiria uma flexibilidade maior para instanciá-los sob demanda, algo que nosso agente procura aprender ao interagir com o ambiente.

Hu e Chen (2013) adotaram aprendizagem por reforço supervisionada para balancear o tráfego. A premissa é que mesmo que se adote como política de encaminhamento o menor caminho entre dois hosts, se muitos pacotes trafegarem pela mesma rota, esta pode ficar congestionada. Dessa forma, outro caminho alternativo, embora mais longo, poderia ser uma alternativa mais adequada. Eles usaram o algoritmo Q-Learning para encaminhar pacotes por caminhos diferentes da rede, com a figura de um supervisor presente para conferir recompensas extras ao agente e otimizar sua aprendizagem, além daquelas provenientes do ambiente, em um cenário de aprendizagem por reforço supervisionado. Para o nosso trabalho, buscou-se adotar aprendizagem de máquina para balancear o tráfego de maneira *online*, sem a necessidade de um supervisor.

Em um trabalho semelhante, Kim et al. (2016) também mesclam SDN e aprendizagem por reforço para tratar congestionamento de fluxo. Os pesquisadores definiram um limiar em cada link, e quando o tráfego o supera, o controlador procura por uma nova rota a partir do algoritmo Q-Learning. O controlador SDN é figura central no reordenamento de rotas. O foco do balanceamento de carga é diminuir o estresse somente sobre os links, sem levar em consideração o host que eventualmente seja o destino de todo o tráfego. Nosso trabalho buscou propor um agente que não somente distribui o tráfego, mas também se necessário instancia réplicas de servidores ao realizar o balanceamento.

6. Conclusão e Trabalhos Futuros

Neste trabalho foi proposto um sistema para detectar e mitigar anomalias em redes definidas por software com auxílio da tecnologia NFV e de técnicas baseadas em aprendizagem por reforço. Propôs-se um agente que reside no orquestrador da arquitetura NFV e coleta evidências de anomalias a partir de métricas de rede. As métricas, além de fornecerem a base para a evidência de anomalias, são utilizadas para a construção de perfis de rede. Os perfis de rede procuram hierarquizar o tratamento a diferentes ameaças, fazendo com que o agente se concentre em eliminar primeiro determinadas anomalias mais urgentes. Tal hierarquização também serve para evitar que uma ação de um perfil destinada a eliminar uma ameaça acabe anulando os efeitos de uma outra ação de outro perfil.

Foi verificado que, tipicamente, estudos publicados na literatura procuram tratar, ainda que com profundidade, soluções para apenas um tipo específico de anomalia de rede. Este trabalho tem como uma das principais contribuições, portanto, procurar combinar diferentes técnicas de mitigação para diferentes anomalias harmonicamente, e para os experimentos realizados, ainda que em ambientes virtualizados, houve resultados promissores. Por outro lado, também por meio da análise de resultados, foi possível constatar que a representação dos estados apresenta um elevado custo de armazenamento: um pequeno incremento no número de switches provoca um grande aumento na quantidade de estados que podem ser visitados.

Por fim, vislumbra-se para o futuro aprimoramentos no sistema aqui apresentado a ampliação do universo de perfis modelados. Por exemplo, incluir monitoramento de escaneamento de portas, sistema detector de intrusões (IDS), entre outros. Também propõe-se o emprego de mecanismos mais sofisticados para detecção de ataques, uma vez que esse não foi o foco do artigo, onde optamos por adotar uma abordagem mais simplista. No entanto, uma alternativa para o futuro seria a adoção de técnicas, por exemplo, baseadas em entropia [da Silva et al. 2016]. Além disso, a adoção de outras técnicas para representação dos estados dos perfis de rede é um tópico a ser investigado. Em vez de usar a abordagem de discretização de valores e posterior armazenamento em tabelas, poderia-se adotar redes neurais ou Tile-Coding [Sutton e Barto 2012]. Por fim, citamos a coleta de outras métricas para a representação dos estados. Por exemplo, o perfil de réplica de servidor poderia levar em consideração não somente a vazão no enlace de acesso do servidor à rede, mas também consumo de CPU e uso de memória.

Referências

B. A. A. Nunes, M. M., Nguyen, X. N., Obraczka, K., and Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys Tutorials*, 16(3):1617–1634.

Belyaev, M. and Gaivoronski, S. (2014). Towards load balancing in sdn-networks during ddos-attacks. In *Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), 2014 International*, pages 1–6.

da Silva, A. S., Wickboldt, J. A., Granville, L. Z., and Schaeffer-Filho, A. (2016). ATLANTIC: A framework for anomaly traffic detection, classification, and mitigation in SDN. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 27–35.

ETSI (2012). Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges & Call for Action. V. 1, 22-24/10/2012.

ETSI (2014). Network Functions Virtualisation (NFV): Network Operator Perspectives on Industry Progress. V. 3, 14-17/10/2014.

Herrera, J. G. and Botero, J. F. (2016). Resource Allocation in NFV: A Comprehensive Survey. *IEEE Transactions on Network and Service Management*, PP(99):1–1.

Hu, Z. and Chen, H. (2013). Network load balancing strategy based on supervised reinforcement learning with shaping rewards. In *Intelligent Control and Information Processing (ICICIP), 2013 Fourth International Conference on*, pages 393–397.

Kim, S., Son, J., Talukder, A., and Hong, C. S. (2016). Congestion prevention mechanism based on Q-learning for efficient routing in SDN. In *2016 International Conference on Information Networking (ICOIN)*, pages 124–128.

King, D., Farrel, A., and Georgalas, N. (2015). The role of SDN and NFV for flexible optical networks: Current status, challenges and opportunities. In *2015 17th International Conference on Transparent Optical Networks (ICTON)*, pages 1–6.

Machado, C. C., Granville, L. Z., and Schaeffer-Filho, A. (2016). ANSwer: Combining NFV and SDN features for network resilience strategies. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 391–396.

Malialis, K. (2014). *Distributed Reinforcement Learning for Network Intrusion Response. University of York*. PhD thesis, United Kingdom.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

ONF (2014). Sdn architecture overview.

Sterbenz, J. P. G., Hutchison, D., Çetinkaya, E. K., Jabbar, A., Rohrer, J. P., Schöller, M., and Smith, P. (2010). Resilience and Survivability in Communication Networks: Strategies, Principles, and Survey of Disciplines. *Comput. Netw.*, 54(8):1245–1265.

Sutton, R. S. and Barto, A. G. (2012). *Introduction to Reinforcement Learning*. MIT Press, Massachusetts, EUA, 2 edition.

Yau, D. K. Y., Lui, J. C. S., Liang, F., and Yam, Y. (2005). Defending Against Distributed Denial-of-service Attacks with Max-min Fair Server-centric Router Throttles. *IEEE/ACM Trans. Netw.*, 13(1):29–42.

NFV em Redes 5G: Avaliando o Desempenho de Composição de Funções Virtualizadas via Maestro

Ariel Galante Dalla-Costa[1], **Matias A. K. Schimuneck**[1], **Juliano Araujo Wickboldt**[1], **Cristiano Bonato Both**[2], **Luciano Paschoal Gaspary**[1], **Lisandro Zambenedetti Granville**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul

[2]Departamento de Matemática Aplicada e Ciências Sociais
Universidade Federal das Ciências da Saúde de Porto Alegre

{agdcosta, makschimuneck, jwickboldt}@inf.ufrgs.br

cbboth@ufcspa.edu.br, {paschoal, granville}@inf.ufrgs.br

Abstract. *Dynamic Cloud Radio Access Network (Dynamic C-RAN) is an architecture wireless that aims benefits such as flexibility and agility to new five generation (5G) network. In Dynamic C-RAN, the radio functions processing is distributed along of a hierarchical of clouds. Network Functions Virtualization has been investigated the facilities to employ this wireless functions. NFV offers orchestration of split of radio functions into Virtualized Network Functions (VNF). The orchestrator needs to consider strict requirements of C-RAN between distributed components of VNF to get real earnings. In this work, we propose a detailed evaluation of Maestro - an NFV orchestrator for wireless environments aware VNF composition. We evaluate Maestro through of linear model programming, considering the effectiveness and orchestration overhead for deciding the best place to VNF components. In addition, we evaluate the trade-off between network data rate and time to generate the placement of components.*

Resumo. *Dynamic Cloud Radio Access Network (Dynamic C-RAN) é uma arquitetura de redes sem fio emergente que objetiva trazer benefícios como flexibilidade e agilidade para redes móveis de quinta geração (5G). Em Dynamic C-RAN, a carga de processamento das funções de rádio é distribuída ao longo de uma hierarquia de nuvens e, para tal, tem-se investigado o uso de Network Function Virtualization (NFV). NFV tem potencial para oferecer a orquestração das funções de rádio, através da sua subdivisão em componentes de Virtualized Network Function (VNF). Neste trabalho propõe-se uma avaliação detalhada do Maestro – um orquestrador de NFV para ambientes sem fio orientado a composição de VNFs, a fim de estimar os potenciais ganhos ao se aplicar os conceitos de NFV em Dynamic C-RAN. Através de um modelo baseado em programação linear, avaliou-se a efetividade e custo de orquestração em termos de aderência aos requisitos de comunicação entre as funções de rádio e os posicionamentos dos componentes das VNFs na rede. Além disso, avaliou-se a relação entre tempo para realizar esses posicionamentos e a taxa de dados transmitida, considerando cenários de ocupação variável em Dynamic C-RAN.*

1. Introdução

Cloud Radio Access Network (C-RAN) é uma arquitetura de redes sem fio que separa os componentes das *Base Stations* (BS) em *Radio Remote Head* (RRH) e *Baseband Unit* (BBU). Uma RRH é uma unidade de rádio, enquanto uma BBU é uma unidade de processamento de sinais [Checko et al. 2015]. As RRHs estão localizadas na RAN e as amostras de sinais são transportadas através de enlaces de fibra óptica até as BBUs, que são centralizadas em uma infraestrutura de nuvem, também conhecidas como BBU *pool*. *Dynamic* C-RAN é uma evolução do conceito original de C-RAN, que prevê a distribuição das funções de rádio sobre uma infraestrutura de nuvem, como por exemplo, uma nuvem de borda que processa parte das funções de rádio, e uma nuvem regional que realiza o processamento das demais funções [5G-PPP-Working-Group 2016]. Em *Dynamic* C-RAN, o processamento das funções de rádio pode ser dividido de acordo com as suas funcionalidades, sendo distribuídas em diferentes locais da infraestrutura da rede, resultando em benefícios, tais como adaptabilidade, balanceamento de carga, flexibilidade na implantação de serviços e eficiência energética [Bartelt et al. 2015].

Apesar dos benefícios de *Dynamic* C-RAN, coordenar as funções de rádio não é uma tarefa trivial, por exemplo distribuir as funções de rádio ao longo de diferentes níveis da rede necessita de um algoritmo especializado. Esse algoritmo de distribuição deve respeitar requisitos restritos, tais como taxa mínima de dados transmitidos e atraso mínimo. Por esta razão, tanto academia quanto indústria vêm explorando conceitos de *Network Functions Virtualization* (NFV), na coordenação e distribuição das funções de rede. Utilizar NFV em *Dynamic* C-RAN, torna possível mover as funções de rádio para uma nuvem distribuída, virtualizando essas funções, chamadas de *Virtualized Network Functions* (VNF). Desta forma, as características de NFV, como por exemplo, escalabilidade, migração e orquestração, podem ser aproveitadas em cenários dinâmicos, como em *Dymamic* C-RAN [Heideker e Kamienski 2016], [Abdelwahab et al. 2016].

Os conceitos de NFV devem ser adaptados para gerenciar e orquestrar as funcionalidades de rádio virtualizadas na próxima geração de redes sem fio [Mijumbi et al. 2016], permitindo adaptação e dinamicidade ao ambiente 5G. Para tal, faz-se necessário calcular o posicionamento das funções, atendendo aos requisitos restritos do ambiente *Dynamic* C-RAN. Mais especificamente, um orquestrador deve determinar a localização dos componentes de funções virtualizadas, chamados de *Virtualized Network Functions Components* (VNFCs) ao longo das *Virtualized Deployment Units* (VDUs), considerando as restrições temporais das aplicações. Esse processo, conhecido como posicionamento de funções, é de responsabilidade do orquestrador, pois uma mesma função pode ser dividida em VNFs de diferentes formas pré-estabelecidas.

O orquestrador, chamado *Maestro* [Dalla-Costa et al. 2017], foi proposto em um trabalho anterior, que considera a composição de funções virtualizadas na implantação de VNFCs ao longo da rede. Naquele trabalho, conduziu-se uma prova de conceito focando nas características funcionais e na simulação de ambientes *Dynamic* C-RAN. Neste artigo, realiza-se uma avaliação detalhada do *Maestro*, utilizando diferentes composições de VNFs, buscando caracterizar a capacidade deste orquestrador em satisfazer requisitos temporais restritos, impostos pelo ambiente dinâmico de rede de acesso sem fio. Ademais, avalia-se a relação entre o tempo consumido para o cálculo de posicionamento dos VNFCs e a redução da quantidade de dados transmitidos (ganho de até 83,68%) no *fronthaul*. As-

sim, a principal contribuição deste artigo é mostrar o posicionamento dos VNFCs em um ambiente *Dynamic* C-RAN, devido a ocupação das BSs, observando quais cenários e quais características de ocupação influenciam no tempo para realizar tal posicionamento.

O restante deste artigo está organizado da seguinte maneira. Na Seção 2 são discutidos os trabalhos relacionados. Na Seção 3 é descrito e exemplificado um cenário de *Dynamic* C-RAN. Na Seção 4, propõe-se um estudo de caso, descrevendo e discutindo o desempenho do orquestrador. Finalmente, na Seção 5 são apresentadas as conclusões e os trabalhos futuros.

2. Trabalhos relacionados

Nessa seção são apresentados os trabalhos relacionados sobre *Dynamic* C-RAN e, posteriormente, discute-se o estado da arte sobre orquestração e posicionamento de funções virtualizadas, através de NFV.

2.1. *Dynamic* C-RAN

Diferente de C-RAN, em que o processamento de funções de rádio é movido para a nuvem centralizada, em *Dynamic* C-RAN funções de rádio podem ser distribuídas entre nuvens organizadas hierarquicamente. Essa distribuição possui diversas vantagens potenciais quando comparada a C-RAN, como maior eficiência energética, diminuição da latência, maior escalabilidade, flexibilidade e adaptabilidade de serviços. Por essa razão, redes 5G estão sendo idealizadas em ambientes *Dynamic* C-RAN [5G-PPP-Working-Group 2016]. Trabalhos encontrados na literatura especializada em redes sem fio [Bartelt et al. 2015, Liu et al. 2015] demonstram quais são as possibilidades e as vantagens de dividir as funções de rádio, por exemplo através da divisão do processamento local e remoto em diferentes funções, tais como MAC, Soft-Bit, RX Data, *Subframe* e *In-phase & Quadrature* (I/Q). Além disso, estudos mostram como essas subdivisões de funções de rádio implicam na taxa de dados transmitida no *fronthaul* [Wubben et al. 2014].

Coordenar *Dynamic* C-RAN por meio de funções de rádio virtualizadas é um tema de pesquisa que está sendo bastante investigado [Abdelwahab et al. 2016, Liu et al. 2015]. Existem trabalhos na literatura que advogam que NFV pode desempenhar um papel importante no gerenciamento e na orquestração de funções de rede sem fio virtualizadas [Mijumbi et al. 2016, Riggio et al. 2015]. Desta forma, *Dynamic* C-RAN pode considerar a virtualização da BBU, utilizando componentes para cada camada de rádio. Entretanto, ao empregar NFV nesse tipo de rede, depara-se com outros dois principais desafios de pesquisa. O primeiro refere-se a como posicionar e orquestrar funções virtualizadas em uma infraestrutura do tipo *Dynamic* C-RAN. O segundo consiste em assegurar desempenho satisfatório para o cálculo do posicionamento de VNFs, de modo a não prejudicar a dinamicidade do ambiente como um todo. Por exemplo, as funções de rádio de retransmissão da camada *Subframe* não suportam latências superiores a 1ms, enquanto na cada MAC o tempo do *Hybrid Automatic Repeat Request* (HARQ) deve ser menor ou igual a 8ms [Ortín et al. 2014]. Outro fator que pode alterar o funcionamento adequado do ambiente *Dynamic* C-RAN é a mobilidade dos *User Equipments* (UEs), ao longo da infraestrutura, exigindo que o orquestrador gerencie a rede de acordo com as variações das demandas do usuário. Na Subseção 2.2 são apresentados os trabalhos relacionados sobre orquestração em um ambiente *Dynamic* C-RAN utilizando NFV.

2.2. Orquestração baseada em NFV

Dando um passo adiante na área de *Dynamic* C-RAN, foi proposta uma arquitetura para NFV em redes 5G [Abdelwahab et al. 2016], discutindo a implantação de funções de rádio virtualizadas. Essa proposta se assemelha a arquitetura *Management and Orchestration* (MANO) do *European Telecommunications Standards Institute* (ETSI) [Quittek et al. 2014]. A arquitetura do ETSI visa padronizar as implementações, fornecer suporte à indústria e prover gerenciamento e orquestração de funções virtualizadas, definindo um *Network Function Virtualization Orchestrator* (NFVO). Segundo a especificação do ETSI, uma VNF é composta por VNFCs, que são os menores elementos da VNF e são implantados em *containers* virtualizados, ou VDUs. Todas essas VNFs são descritas por um VNF *Descriptor* (VNFD), que representa como os VNFCs são distribuídos e implantados ao longo dos VDUs.

Na área de orquestração em NFV, destacam-se, inicialmente, dois trabalhos relacionados a solução de orquestração. O primeiro refere-se a uma plataforma aberta para NFV, baseado em nuvem, que suporta uma API e outros componentes provenientes do *framework* ETSI MANO [Makaya et al. 2015]. O segundo propõe uma arquitetura modular para NFV, que permite o gerenciamento baseado em políticas de VNFs [Giotis et al. 2015]. Além disso, pode-se destacar outros trabalhos que abordam o posicionamento de funções virtualizadas, tais como T-NOVA [Xilouris et al. 2014] e Cloud4NFV [Soares et al. 2014]. Entretanto, nenhum desses trabalhos considera a composição das funções para a tomada de decisão em um ambiente sem fio, sendo assim, não atendem os requisitos necessários para orquestrar ambientes *Dynamic* C-RAN.

Um orquestrador específico para WLAN foi proposto [Riggio et al. 2015], sem considerar a divisão de componentes de funções, ou seja, não sendo adaptável ao cenário de *Dynamic* C-RAN. Além disso, uma avaliação formal de posicionamento de recursos de funções virtualizadas foi proposta [Moens e Turck 2014], focando em cenários híbridos entre VNFs e funções físicas legadas. Em relação as propostas de posicionamento automático dos nós virtuais e alocação de serviços [Clayman et al. 2014], pode-se destacar a abordagem baseada em heurísticas de *Service Function Chaining* (SFC) [Luizelli et al. 2015, Li e Qian 2016]. Essa abordagem, apesar de considerar o posicionamento de VNFs, não leva em consideração os componentes na tomada de decisão, assim, os tempos para efetuar o posicionamento não são diretamente comparáveis. Desta forma, com a finalidade de avaliar a dinamicidade da orquestração em *Dynamic* C-RAN, neste trabalho avalia-se a arquitetura do orquestrador Maestro, proposto anteriormente [Dalla-Costa et al. 2017].

3. Orquestração em *Dynamic* C-RAN

O cenário *Dynamic* C-RAN é composto de uma hierarquia em nuvens, aproveitando a distribuição geográfica das RANs como em ambientes FOG [Bonomi et al. 2012]. A nuvem central geralmente possui a maior quantidade de recursos disponíveis, enquanto as nuvens regionais e de borda são posicionadas próximas ao *front-end*, possuindo geralmente menos recursos. Essa organização hierárquica permite a implantação de funções virtualizadas de forma dinâmica, tornando o provisionamento do serviço mais flexível [Peng et al. 2014]. Por exemplo, uma nova unidade de processamento de sinal virtualizado pode ser implantada em uma nuvem regional para suprir uma determinada demanda,

devido ao aumento do número de usuários na RAN. Além disso, algumas técnicas de balanceamento de carga e de economia de energia tornam-se facilmente aplicáveis nesses cenários, uma vez que existe flexibilidade na migração de VNFs entre os diferentes níveis de nuvem. Assim, dividir e posicionar as funções de rádio que demandam maior transmissão de dados, faz-se necessário através de um orquestrador. Desta forma, na Subseção 3.1 apresenta-se um modelo de posicionamento de VNFs e sua implementação, no orquestrador *Maestro* através de programação linear, é discutida na Subseção 3.2.

3.1. Modelo de posicionamento de VNFs em *Dynamic* C-RAN

Considerando um ambiente *Dynamic* C-RAN, contendo um conjunto de A BSs com um total de N nuvens (borda, regionais e centrais) e capacidade de F funções de rádios virtualizadas, o total de dados transmitidos no *fronthaul* pode ser escrito como:

$$O_B(d) = \sum_{f=0}^{F} \sum_{n=0}^{N} \sum_{a=0}^{A} d_{f,n,a} D_B(f,n,a) \tag{1}$$

onde, $f \in \{0,...,F\}$ corresponde as funções de rádio virtualizadas, $n \in \{0,...,N\}$ representa as nuvens disponíveis para processamento dessas funções e $a \in \{0,...,A\}$ simboliza as BSs correspondentes a uma *Dynamic* C-RAN. A matriz tridimensional $d = [d_{f,n,a}]_{F \times N \times A}$ representa um posicionamento factível para as funções de rádio virtualizadas em cada BSs entre as nuvens. A matriz de posicionamento $d_{f,n,a} \in \{0,1\}$, *i.e.,* $d_{f,n,a} = 1$ indica que f foi posicionada sobre n para processar o sinal de a. Caso contrário, *i.e.,* $d_{f,n,a} = 0$, representa que f da BS a não está sendo processada em n. Finalmente, $D_B(f,n,a)$ determina a taxa de dados necessária para processar f da BS a em uma determinada nuvem n. A equação $D_B(f,n,a)$ considera o tipo de f a ser processada, a distância da nuvem n (*i.e.,* borda, regionais ou centrais) e a demanda de UEs na BS a, para determinar a quantidade de dados transmitidos no *fronthaul*. Maiores informações sobre o cálculo da quantidade de dados transmitidos, podem ser encontrados em Wubben, et al. [Wubben et al. 2014].

Baseado nas definições acima, o problema de minimização do total de dados transmitidos no *fronthaul*, através do posicionamento dinâmico das funções de rádio virtualizadas e suas restrições pode ser formulada como:

$$\min_{\{\mathbf{d}\}} \quad O_B(d) = \min_{\{\mathbf{d}\}} \sum_{f,n,a} d_{f,n,a} D_B(f,n,a) \tag{2}$$

s.t.

$$\sum_{n} d_{f,n,a} = 1, \qquad \forall\, f, \forall\, a, \tag{3}$$

$$\sum_{f,a} d_{f,n,a} D_V(f,n,a) \leq C_V^n, \qquad \forall\, n, \tag{4}$$

$$\sum_{f,a} d_{f,n,a} D_B(f,n,a) \leq C_B^n, \qquad \forall\, n, \tag{5}$$

onde, $D_V(f,n,a)$ determina o consumo dos recursos de nuvem necessários para processar f da BS a em uma determinada nuvem n. C_V^n e C_B^n representam os limites de VDUs e de banda do *fronthaul* da nuvem n, respectivamente.

As restrições (3), (4) e (5) garantem a factibilidade da solução perante o ambiente de uma *Dynamic* C-RAN. A restrição (3) garante que para toda e qualquer BS $a \in A$ exista obrigatoriamente uma função de rádio virtualizada $f \in F$. Além disso, a restrição garante que nenhuma f de uma mesma a seja duplicada dentre todas as nuvens possíveis $n \in N$. A restrição (4) garante que o posicionamento de uma função de rádio virtualizada f de uma BS a seja realizada somente em uma nuvem n, com capacidade de processamento, *i.e.,* a nuvem deve conter VDUs suficientes para atender todas as funções atribuídas. Por fim, a restrição (5) garante que o posicionamento de uma função f de uma BS a ocorra obrigatoriamente em uma nuvem n com capacidade de transmissão, *i.e.,* uma nuvem só pode receber f caso o *fronthaul* possua banda suficiente para atender os requisitos de transmissão de todas as funções virtualizadas nessa nuvem.

3.2. Programação linear dos recursos em *Dynamic* C-RAN utilizando Maestro

O diferencial proposto pelo orquestrador *Maestro* em relação aos trabalhos anteriores é a capacidade de posicionar VNFCs em diferentes VDUs, sobre diferentes nuvens, quando as funções virtualizadas possuem mais que uma opção de divisão. Sendo assim, o *Maestro* pode posicionar os VNFCs de várias maneiras diferentes, possibilitando ganhos na transmissão de dados e na maximização do uso dos recursos computacionais nas nuvens de borda. Desta forma, o orquestrador recebe um conjunto de informações sobre a infraestrutura de rede, definidas pelo operador, e sua utilização, para determinar o posicionamento de cada função virtualizada para todas as BSs dentre as nuvens existentes.

A principal função do *Maestro* é minimizar a taxa de dados transmitidos no *fronthaul*, adotando o modelo descrito na Subseção 3.1. Para acelerar a resolução do problema linear, um novo elemento objetivo é inserido em (2), para maximizar a quantidade de recursos utilizados nas nuvens de borda, uma vez que estas nuvens não transmitem dados no *fronthaul* e devem ser priorizas pelas soluções. Desta maneira, insere-se uma ponderação na função objetivo, afim de atingir os objetivos propostos. Tal minimização ponderada é descrita como:

$$\begin{aligned} \min_{\{\mathbf{d}\}} \quad & \omega \cdot \sum_{f,n,a} d_{f,n,a} D_B(f,n,a) + (\omega - 1) \cdot \sum_{f,a} d_{f,0,a} D_V(f,n,a) \\ \text{s.t.} \quad & (3)(4)(5) \end{aligned} \tag{6}$$

onde, $0 \leq \omega \leq 1$ realiza a ponderação das funções objetivos para resolução do problema e $d_{f,0,a} D_V(f,n,a)$ da prioridade das nuvens de borda no posicionamento das funções virtualizadas. A segunda parte de (6) tem por finalidade maximizar a ocupação de recursos nas nuvens de borda e acelerar a convergência para a solução. A maximização das nuvens de borda e a minimização da taxa de dados transmitidos no *fronthaul* fazem com que os algoritmos consigam obter um melhor desempenho quanto ao tempo, através do direcionamento das possibilidades de posicionamento das VNFs, durante sua execução.

O *Maestro* garante que os componentes de funções de rádio não sejam implantados na mesma nuvem para a mesma BS, que não se exceda a capacidade computacional das nuvens e a capacidade de transmissão disponível no *fronthaul*, respeitando as restrições descritas anteriormente. O *Maestro* foi implementado utilizando o *IBM CPLEX* [IBM Software 2010], com a função objetivo descrita em (6). Na Seção 4 é apresentado o estudo de caso e os experimentos utilizados para avaliação do *Maestro*.

4. Estudo de Caso e Experimentos

O estudo de caso e os experimentos apresentados nesta seção visam demonstrar o desempenho do orquestrador *Maestro* em posicionar as composições de funções mais apropriadas para a implantação dos componentes de VNFs. Baseado em uma infraestrutura hierárquica, modelou-se um conjunto de VDUs, juntamente com um conjunto de BSs, que são descritos a seguir.

4.1. Estudo de Caso

Este estudo de caso é baseado nas opções de divisão de funções de rádio, descritas por Wubben, et al. [Wubben et al. 2014] e aplicadas no ambiente de *Dynamic* C-RAN. A abordagem proposta por Wubben, et al. prevê a execução distribuída de cinco funções de rádio, a saber: *MAC*, *Soft-Bit*, *RX Data*, *Subframe* e *I/Q*. Essa abordagem objetiva permitir um posicionamento mais flexível dos recursos de computação e de rede necessários para processar o sinal de rádio de forma distribuída, através do uso de virtualização. Maiores detalhes sobre a operação de cada divisão e das próprias funções de rádio são descritas em Wubben, et al. [Wubben et al. 2014]. A decisão acerca da utilização de cada opção de divisão gera uma variabilidade na taxa de dados transmitidos no *fronthaul* e de recursos de computação nas nuvens, o que torna essa abordagem interessante para este estudo de caso. Considerando essa abordagem, é possível mostrar como a orquestração de componentes de VNFs causa impacto no desempenho da rede e na utilização de recursos.

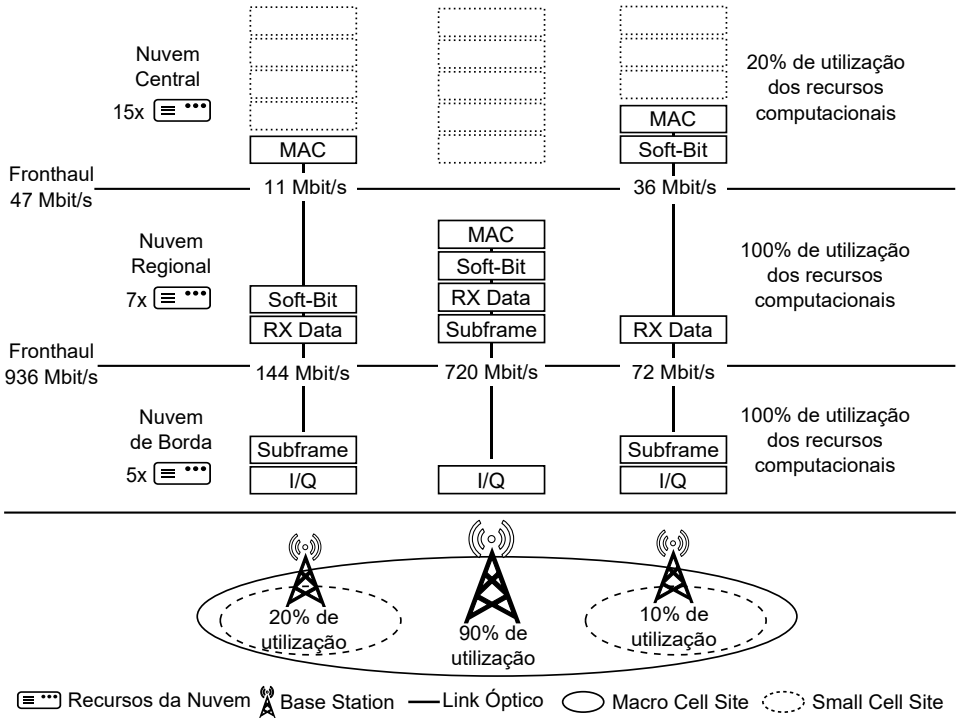


Figura 1. Exemplo de divisões de funções de rádio implantadas como VNFs

Na Figura 1 pode-se observar um exemplo de posicionamento das funções de rádio em componentes de VNFs e as suas respectivas distribuições ao longo dos níveis de nuvens disponíveis (*i.e.*, borda, regional e central). No exemplo, cada VNF implementa uma BS, sendo que cada uma das cinco funções de rádio são encapsuladas em um VNFC,

que por sua vez são implantados em até 3 VDUs. As BSs podem ser visualizadas na parte inferior da Figura 1, onde cada uma delas possui uma porcentagem de ocupação diferente em função da capacidade total de transmissão. Essa variação de ocupação influencia o impacto das opções de divisão selecionadas em termos de dados transmitidos no *fronthaul*. No exemplo, a *macro cell* na parte inferior central, tendo ocupados 90% de sua capacidade, foi implantada com 2 VDUs posicionados na borda e na nuvem regional, consumindo uma taxa fixa de 720Mbit/s para comunicação entre as funções de rádio I/Q e *Subframe*. Já para as *small cells*, tendo ocupações de apenas 10% e 20%, optou-se por utilizar 3 VDUs posicionando as funções acima do RX Data – que têm seus requisitos de banda minimizados pelas baixas taxa de ocupação – nas nuvens regional e central.

A lógica por trás do exemplo da Figura 1 é de tentar posicionar os VDUs de forma a minimizar o total de dados transmitidos no *fronthaul*. Consequentemente, a nuvem de borda, que fica mais próxima da RAN, é prioritariamente ocupada com funções de rádio de nível mais baixo que são aquelas que apresentam requisitos mais estritos de comunicação. A nuvem regional, por sua vez, possui maior quantidade de recursos computacionais em relação a nuvem de borda, porém encontra-se mais distante da RAN. Sendo assim, as funções de rádio implantadas na nuvem regional enfrentam atrasos de comunicação para transporte dos dados até esse local. Finalmente, a nuvem central é a nuvem que possui maior quantidade de recursos disponíveis, porém é também aquela que está mais distante da RAN. Essa nuvem se torna mais atraente para o processamento de funções de nível mais alto (*e.g.*, Soft-Bit ou MAC), principalmente para BSs com menor ocupação. Vale salientar que, por medida de simplificação, nesse exemplo considera-se que cada função de rádio encapsulada em um VNFC consome uma unidade de processamento de uma nuvem onde é posicionada. Em um ambiente real, a demanda de processamento potencialmente seria variável em função de fatores como a ocupação das BSs, por exemplo.

Sabendo que os dados transmitidos no *fronthaul* variam de acordo com a ocupação das BSs, para realizar os experimentos foi necessário estimar os percentuais de ocupação com base em uma simulação utilizando as recomendações do *3rd Generation Partnership Project* (3GPP), do documento *Further advancements for E-UTRA physical layer aspects* [3GPP 2010]. Esse percentual de ocupação foi calculado através do método de *Monte Carlo*, em uma simulação do tráfego gerado por UEs associados a uma infraestrutura com 7 *macro cells* e 15 *small cells* distribuídas dentro da área de cobertura de cada *macro cell*, totalizando 112 BSs. Em média, 60 usuários foram distribuídos aleatoriamente dentro da área de cobertura de cada *macro cell* e associados a BS mais próxima de cada usuário (seja uma *macro* ou *small cell*). A distribuição das BSs e UEs foi realizada em uma área de 2000x2000 metros, seguindo as normas para simulação definidas no documento *Small cell enhancements for E-UTRA and E-UTRAN - Physical layer aspects* do 3GPP [3GPP 2013] para uma rede celular, considerando a existência de áreas com diferentes densidades de BS e UEs *i.e.,* zonas rurais e urbanas.

Na Tabela 1 pode-se encontrar a ocupação das BSs utilizadas na simulação, expressa através de uma distribuição discreta de probabilidades, servindo como entrada para o modelo linear detalhado na Seção 3. A ocupação das BSs influencia a taxa de dados transmitidos para as funções de *MAC, Soft-Bit, RX Data*, tal como é descrito por Wubben et al. [Wubben et al. 2014]. A quantidade de *macro cells* consideradas na simulação cor-

Tabela 1. Distribuição da Ocupação das *macro* e *small cells*

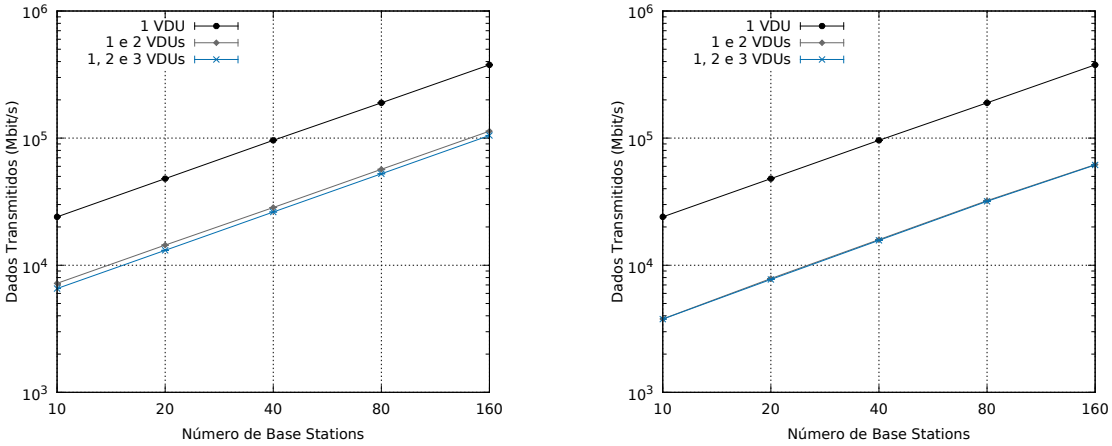
	Macro cells		*Small cells*	
Índice de Ocupação	Carga de Ocupação	Probabilidade de Ocupação	Carga de Ocupação	Probabilidade de Ocupação
0	15%	0,42%	2,5%	56,97%
1	25%	1,42%	12,5%	31,59%
2	35%	5,14%	22,5%	9,28%
3	45%	10,28%	37,5%	1,88%
4	65%	15,71%	47,5%	0,28%
5	75%	19,17%	-	-
6	85%	17,29%	-	-
7	95%	30,57%	-	-
	Total	100%	-	100%

responde a 33,3% da infraestrutura, enquanto que as *small cells* representam 66,7%. Essa medida se deve à quantidade de UEs distribuídos ao longo da região geográfica, sendo maior para as *macro cells* e menor para as *small cells*. Os cenários de avaliação possuem 10, 20, 40, 80 e 160 BSs ao longo da infraestrutura. Para esse trabalho, considerou-se que cada um dos três níveis hierárquicos de nuvem possui respectivamente 5, 7 e 15 unidades de processamento, como pode ser observado na Figura 1, aumentando essas unidades proporcionalmente à quantidade de BSs em cada cenário. Baseado os cenários de simulação apresentados, na Subseção 4.2 os resultados obtidos são descritos e analisados.

4.2. Resultados

O modelo de programação linear foi implementado com *IBM CPLEX Optimizer* [IBM Software 2010], utilizando 8 *threads* e o mesmo algoritmo *Dual Simplex* para todos os cenários. A simulação foi executada em um computador com um processador Intel i7 de 3.1 GHz, com 16 GB de memória RAM, executando sobre um sistema operacional Linux de 64 *bits*. Para todos os cenários, considerou-se três nuvens (*i.e.*, de borda, regional e central), com capacidade proporcional para acomodar a carga de trabalho nos diferentes cenários, tal como descrito na Subseção 4.1. Todos os experimentos foram repetidos 50 vezes e o tempo para posicionar os VNFCs ao longo dos VDUs, foi medido pelo *IBM CPLEX*. Neste artigo, considerou-se tanto a ocupação fixa das BSs no posicionamento das funções de rádio, quanto ocupação variável (conforme a Tabela 1), diferentemente da avaliação realizada no trabalho anterior [Dalla-Costa et al. 2017]. Desta forma, é possível comparar a quantidade de tempo necessária para efetuar o posicionamento das funções de rádio em ambos os casos.

Pode-se observar na Figura 2(a) a variação total de dados transmitidos no *fronthaul* para diferentes opções de divisão de funções de rádio, considerando que as BSs tenham ocupação percentual fixa em 50%. A Figura 2(b) apresenta o mesmo resultado, porém considerando ocupação variável das BSs (aleatoriamente atribuídos conforme a Tabela 1). O eixo x denota a quantidade de BSs em cada experimento, enquanto o eixo y (em escala logaritmica) denota os dados transmitidos no *fronthaul* em Mbit/s ocasionado pelas opções de divisão selecionadas pela função de posicionamento. A curva *1 VDU* representa a transmissão dos dados considerando uma VNF atômica, utilizando apenas um VDU contendo todos os VNFCs. Já a curva *1 e 2 VDUs* expressa a transmissão dos dados



(a) Percentual de ocupação fixa das BSs (b) Percentual de ocupação variável das BSs

Figura 2. Transmissão de dados no *fronthaul*

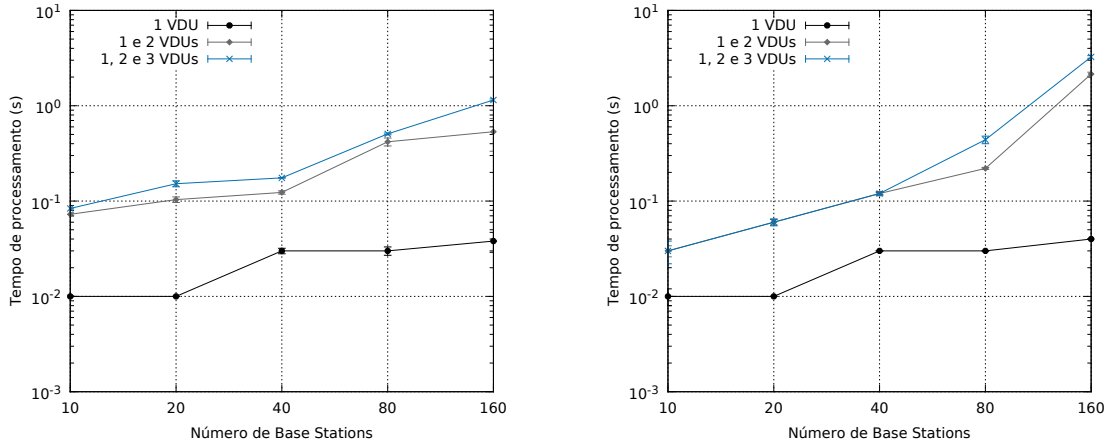
quando VNFs são posicionadas em um ou dois VDUs (*i.e.*, podendo se utilizar uma opção de divisão para separar VNFCs em até dois VDUs). Por fim, a curva *1, 2 e 3 VDUs* representa os dados transmitidos de uma VNF implantada em até 3 VDUs, similar ao que é demonstrado na Figura 1 (*i.e.*, podendo usar simultaneamente até duas opções de divisão para separar os VNFCs de uma mesma VNF).

Observando o gráfico de dados transmitidos com ocupação fixa, para 10 BSs a quantidade de dados transmitido utilizando apenas 1 VDU foi de 24 Gbit/s, enquanto que até 2 e até 3 VDUs esse valor é reduzido para 7,2 Gbit/s e 6,5 Gbit/s, respectivamente. Percentualmente, houve uma redução de 70% entre os cenários utilizando 1 VDU versus 2 VDUs e de 72,7% comparando 1 e 3 VDUs. Entre os cenários de até 2 e até 3 VDUs a diferença foi apenas de 9%. Já no caso mais extremo, considerando 160 BSs, a banda consumida utilizando 1 VDU atingiu 376,8 Gbit/s, enquanto que para até 2 VDUs foi de 113,4 Gbit/s e para até 3 VDUs foi de 104,83 Gbit/s. Assim, manteve-se aproximadamente os mesmos preceituais de redução entre as curvas, enfatizando a importância da composição de VNFs no uso racional dos recursos do *fronthaul*.

Ao observar o gráfico de dados transmitidos com ocupação variável para 10 BSs, a quantidade de dados transmitidos para 1 VDU foi dos mesmos 24 Gbit/s obtidos com ocupação fixa. Isso se deve ao fato de que no posicionamento de 1 VDU pode-se utilizar apenas a opção de divisão da função de rádio I/Q, que requer uma taxa de transferência fixa independente da ocupação da BS. Ainda considerando 10 BSs, as curvas de até 2 VDUs e até 3 VDUs obtiveram taxas de transmissão de dados na ordem de 3,8 Gbit/s e 3,7 Gbit/s, respectivamente, atingindo reduções de aproximadamente 83% em comparação com a curva de 1 VDU. Na comparação dos resultados entre as curvas de 2 VDUs e 3 VDUS, os valores se mantiveram sempre muito similares para todas as quantidade de BSs simuladas com diferenças sempre abaixo de 1%.

Neste trabalho, diferentemente dos trabalhos anteriores, compara-se o desempenho do posicionamento de BSs considerando ocupação fixa e variável em ambientes de *Dynamic* C-RAN. Pode-se observar que tanto na ocupação fixa como na variável, o orquestrador é capaz de minimizar a transmissão de dados no *fronthaul*, quando existir a

possibilidade de selecionar as divisões de funções de rádio com 2 ou 3 VDUs. Essa redução é ainda mais evidente quando há demanda variável nas BSs, o que se deve principalmente a dois fatores: (*i*) existe uma diferença de ocupação média significativa entre *macro* e *small cells*, o que fornece ao orquestrador uma gama de opções de VDUs com transmissão de dados diferentes para posicionar; e (*ii*) apesar da alta ocupação nas *macro cells* (em média 75%), há um grande número de *small cells* com baixa ocupação (em média 8%), o que reduz a ocupação média por BS para a casa dos 30%. A utilização de até 3 VDUs, ainda que acarrete ganhos em relação a até 2 VDUs, traz vantagens muito tímidas no cenário de ocupação variável, o que coloca em dúvida se de fato o vale esforço extra para calcular posicionamentos mais complexos para obter ganhos pouco significativos.



(a) Solução na ocupação fixa

(b) Solução na ocupação variável

Figura 3. Tempo necessário para encontrar uma solução

As Figuras 3(a) e 3(b) ilustram o tempo médio para computar o posicionamento de todos os VNFCs sobre a infraestrutura de nuvens disponível, considerando as BSs com ocupação fixa e variável. As barras de erro representam um intervalo de confiança de 99%, obtido por meio de 50 repetições para cada experimento. O eixo x expressa o número de BSs no cenário, enquanto o eixo y denota o tempo médio em segundos necessário para calcular o posicionamento dos VNFCs.

Os gráficos de tempo demonstram que para até 80 BSs, o tempo necessário para realizar o posicionamento com ocupação fixa é superior ao tempo de ocupação variável. Isso se justifica pois no cenário de ocupação fixa os recursos se esgotam mais rapidamente, de forma que o orquestrador testa mais possibilidades para alocar os VDUs de forma eficiente. Entretanto, para 160 BSs o cenário de ocupação variável consome mais tempo (3,19s para até 3 VDUs, contra 1,12s da ocupação fixa) do que a ocupação fixa. Fato este que pode ser explicado pela grande quantidade de combinações de VDUs com demandas de banda diferentes disponíveis para posicionamento pelo orquestrador.

No cenário de ocupação variável, para realizar o posicionamento de até 40 BSs utilizando 2 ou 3 VDUs, os tempos de cálculo são extremamente similares. Isso torna viável a utilização de divisões de funções de rádio em até 3 VDUs, uma vez que isso leva a uma redução, ainda que pequena, no consumo de recursos do *fronthaul*. Já para 80 e 160

BSs, o tempo necessário para realizar o posicionamento utilizando até 3 VDUs, ocasiona um aumento de mais de aproximadamente 40% no tempo médio de calculo em relação a até 2 VDUs. Desta forma, pode-se concluir que em um ambiente com uma grande quantidade de BSs, onde a variação da ocupação for frequente, a utilização de até 2 VDUs pode ser mais adequada para permitir que o orquestrador responda mais rapidamente às mudanças do ambiente sem prejudicar significativamente o consumo de recursos.

Finalmente, vale ressaltar que não está sendo considerado neste trabalho o tempo necessário para a implantação/migração dos componentes de VNFs entre as nuvens, o que, em uma infraestrutura real, deverá influenciar no tempo total necessário para que a nova configuração calculada pelo orquestrador entre em vigor. Ademais, os tempos de resposta da modelagem linear sempre são dados até a execução total do posicionamento dos VDUs, de acordo com a função objetivo definida na Seção 3.2. Nesse caso, se essa função for modificada, os tempos de posicionamento serão diferentes.

5. Conclusão e Trabalhos Futuros

Neste trabalho, foi realizada uma avaliação do *Maestro*, um orquestrador para ambientes *Dynamic* C-RAN, que baseia-se nos componentes de funções virtualizadas para realizar o posicionamento das VNFs. Seu mecanismo de decisão foi baseado na modelagem de posicionamento de VNFs e em programação linear dos recursos em *Dynamic* C-RAN. A avaliação foi realizada baseada em simulação de um caso de uso, considerando as especificações de uma rede celular do 3GPP. O *Maestro* cumpre com o objetivo proposto, reduzindo a taxa de dados transmitida no *fronthaul* e maximizando a quantidade de recursos utilizados nas nuvens de borda e regionais, tentando maximizar o uso dos recursos computacionais nas nuvens de borda. Ademais, o *Maestro* fornece uma visão geral dos ganhos que podem ser adquiridos através da divisão de funções de rádio em diferentes nuvens e VDUs.

Além disso, pode-se concluir que se o mecanismo de decisão necessita ser executado a cada 3,19 segundos (pior tempo para 160 BSs), consegue-se atender os requisitos do ambiente de uma rede *Dynamic* C-RAN, encontrando uma solução de posicionamento de todos os componentes. Vale ressaltar que quando implantado em uma rede física, faz-se necessário considerar outros parâmetros, tal como a quantidade de tempo para a migração de uma VNF de um VDU para outro, e desta forma, o mecanismo de decisão pode ter seu tempo para calcular o posicionamento, alterado.

Como trabalho futuro pretende-se implantar o *Maestro* em uma infraestrutura real, conduzindo experimentação de forma real, utilizando elementos do *Maestro* para realizar o posicionamento dos componentes ao longo da rede. Além disso, pretende-se adaptar o orquestrador *Maestro* para outros tipos de rede, como por exemplo, para redes ópticas. Outro caminho a seguir, é que o *Maestro* possa considerar fatores (*i.e.* políticas) durante o processo de decisão, de acordo com regras pré-definidas por operadores de rede.

Agradecimentos

Este trabalho foi financiado pelo programa Horizon 2020 da União Europeia para pesquisa, desenvolvimento tecnológico e demonstração no âmbito do acordo nº. 688941 (FUTEBOL), bem como pelo Ministério da Ciência, Tecnologia, Inovação e Comunicação (MCTIC) através da RNP/CTIC.

Referências

3GPP (2010). Further advancements for E-UTRA physical layer aspects; Evolved Universal Terrestrial Radio Access - TR 36.814. Technical report. Disponível em: $www.qtc.jp/3GPP/Specs/36814-900.pdf$.

3GPP (2013). Small cell enhancements for E-UTRA and E-UTRAN - Physical layer aspects; Technical Specification Group Radio Access Network - TR 36.872. Technical report. Disponível em: $www.3gpp.org/ftp/Specs/archive/36\_series/36.872/36872-c10.zip$.

5G-PPP-Working-Group (2016). View on 5G Architecture. Technical report, 5G PPP.

Abdelwahab, S., Hamdaoui, B., Guizani, M., e Znati, T. (2016). Network function virtualization in 5G. *IEEE Communications Magazine*, 54(4):84–91.

Bartelt, J., Rost, P., Wubben, D., Lessmann, J., Melis, B., e Fettweis, G. (2015). Fronthaul and backhaul requirements of flexibly centralized radio access networks. *IEEE Wireless Communications*, 22(5):105–111.

Bonomi, F., Milito, R., Zhu, J., e Addepalli, S. (2012). Fog Computing and Its Role in the Internet of Things. In *SIGCOMM Workshop on Mobile Cloud Computing (MCC)*, MCC '12, pages 13–16, New York. ACM.

Checko, A., Christiansen, H. L., Yan, Y., Scolari, L., Kardaras, G., Berger, M. S., e Dittmann, L. (2015). Cloud RAN for Mobile Networks - A Technology Overview. *IEEE Communications Surveys Tutorials*, 17(1):405–426.

Clayman, S., Maini, E., Galis, A., Manzalini, A., e Mazzocca, N. (2014). The dynamic placement of virtual network functions. In *IEEE Network Operations and Management Symposium (NOMS)*, pages 1–9.

Dalla-Costa, A. G., Bondan, L., Wickboldt, J. A., Both, C. B., e Granville, L. Z. (2017). Maestro: An nfv orchestrator for wireless environments aware of vnf internal compositions (to appear). *IEEE International Conference on. Advanced Information Networking and Applications*.

Giotis, K., Kryftis, Y., e Maglaris, V. (2015). Policy-based orchestration of NFV services in Software-Defined Networks. In *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, pages 1–5.

Heideker, A. e Kamienski, C. (2016). Gerenciamento flexível de infraestrutura de acesso público à internet com nfv. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*.

IBM Software (2010). Efficient modeling with the IBM ILOG CPLEX Optimization Studio. Technical Report WSW14059-USEN-02, ILOG Optimization and Analytical Decision Support Solutions, Somers, NY. White Paper.

Li, X. e Qian, C. (2016). A survey of network function placement. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 948–953.

Liu, J., Zhou, S., Gong, J., Niu, Z., e Xu, S. (2015). Graph-based framework for flexible baseband function splitting and placement in C-RAN. In *IEEE International Conference on Communications (ICC)*, pages 1958–1963.

Luizelli, M. C., Bays, L. R., Buriol, L. S., Barcellos, M. P., e Gaspary, L. P. (2015). Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 98–106.

Makaya, C., Freimuth, D., Wood, D., e Calo, S. (2015). Policy-based NFV management and orchestration. In *Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on*, pages 128–134.

Mijumbi, R., Serrat, J., l. Gorricho, J., Latre, S., Charalambides, M., e Lopez, D. (2016). Management and orchestration challenges in network functions virtualization. *IEEE Communications Magazine*, 54(1):98–105.

Moens, H. e Turck, F. D. (2014). VNF-P: A model for efficient placement of virtualized network functions. In *10th International Conference on Network and Service Management (CNSM) and Workshop*, pages 418–423.

Ortín, J., Caballero, P., IMDEA, Rost, P., e NEC (2014). D5.2 final definition of ijoin requirements and scenarios. Technical report, INFSO-ICT-317941 iJOIN, http://www.ict-ijoin.eu/wp-content/uploads/2012/10/D5.2.pdf.

Peng, M., Li, Y., Jiang, J., Li, J., e Wang, C. (2014). Heterogeneous cloud radio access networks: a new perspective for enhancing spectral and energy efficiencies. *IEEE Wireless Communications*, 21(6):126–135.

Quittek, J. et al. (2014). Network Functions Virtualisation (NFV) - Management and Orchestration. White paper, ETSI NFV ISG.

Riggio, R., Rasheed, T., e Narayanan, R. (2015). Virtual network functions orchestration in enterprise WLANs. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1220–1225.

Soares, J., Dias, M., Carapinha, J., Parreira, B., e Sargento, S. (2014). Cloud4NFV: A platform for Virtual Network Functions. In *IEEE International Conference on Cloud Networking (CloudNet)*, pages 288–293.

Wubben, D., Rost, P., Bartelt, J. S., Lalam, M., Savin, V., Gorgoglione, M., Dekorsy, A., e Fettweis, G. (2014). Benefits and Impact of Cloud Computing on 5G Signal Processing: Flexible centralization through cloud-RAN. *IEEE Signal Processing Magazine*, 31(6):35–44.

Xilouris, G., Trouva, E., Lobillo, F., Soares, J. M., Carapinha, J., McGrath, M. J., Gardikis, G., Paglierani, P., Pallis, E., Zuccaro, L., Rebahi, Y., e Kourtis, A. (2014). T-NOVA: A marketplace for virtualized network functions. In *European Conference on Networks and Communications (EuCNC)*, pages 1–5.

NFV-PEAR: Posicionamento e Encadeamento Adaptativo de Funções Virtuais de Rede

Gustavo Miotto[1], **Marcelo Caggiani Luizelli**[1]
Weverton Luis da Costa Cordeiro [1], **Luciano Paschoal Gaspary**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{gmiotto, mcluizelli, wlccordeiro, paschoal}@inf.ufrgs.br

Resumo. *O projeto de mecanismos flexíveis e eficientes para o posicionamento e encadeamento de funções virtualizadas de rede (VNFs) é central para o sucesso de Virtualização de Funções de Rede (*Network Function Virtualization*, NFV). As soluções existentes, no entanto, consideram custos fixos (e imutáveis) de processamento de fluxos e de largura de banda ao posicionar as VNFs em Pontos de Presença da Rede (N-PoPs). Essa limitação torna-se crítica em redes NFV com fluxos cujos comportamentos são altamente dinâmicos e nas quais os requisitos de processamento e os recursos disponíveis nos N-PoPs mudam constantemente. Para preencher essa lacuna é apresentado o NFV-PEAR, um* framework *para o posicionamento e encadeamento adaptativo de VNFs. O NFV-PEAR visa a (re)organizar periodicamente os posicionamentos e encadeamentos de VNFs previamente determinados, objetivando-se manter um desempenho fim-a-fim aceitável mesmo durante flutuações nos custos de processamento e nos requisitos dos fluxos. Paralelamente, busca-se minimizar as mudanças na rede (por exemplo, a realocação de VNFs ou de fluxos) realizadas para cumprir esse objetivo. Os resultados obtidos, a partir de uma avaliação experimental, mostram que o NFV-PEAR tem potencial para reduzir significativamente o número de mudanças na rede necessárias para assegurar o desempenho fim-a-fim esperado para os fluxos, garantindo assim o funcionamento estável dos serviços.*

Abstract. *The design of flexible and efficient mechanisms for proper placement and chaining of virtual network functions (VNFs) is key for the success of Network Function Virtualization (NFV). State-of-the-art solutions, however, consider fixed (and immutable) flow processing and bandwidth requirements when placing VNFs in the Network Points of Presence (N-PoPs). This limitation becomes critical in NFV-enabled networks having highly dynamic flow behavior, and in which flow processing requirements and available N-PoP resources change constantly. To bridge this gap, we present NFV-PEAR, a framework for adaptive VNF placement and chaining. In NFV-PEAR, one may periodically (re)arrange previously determined placement and chaining of VNFs, with the goal of maintaining acceptable end-to-end flow performance despite fluctuations of flow processing costs and requirements. In parallel, NFV-PEAR seeks to minimize network changes (e.g., reallocation of VNFs or network flows) done to fulfill this goal. The results obtained from an experimental evaluation provide evidence that NFV-PEAR has potential to significantly reduce the number of network changes required to ensure end-to-end flow performance, thus ensuring stable operation of network services.*

1. Introdução

Virtualização de Funções de Rede (*Network Function Virtualization*, NFV) é um paradigma relativamente novo que visa migrar funções tradicionalmente executadas por *hardware* especializado (*middleboxes*) para implementações centradas em *software*, rodando em instâncias de máquinas virtuais. Exemplos de funções incluem *firewall*, balanceamento de carga, *proxy*, detecção de intrusão, entre outras. Essa migração leva a vários benefícios, entre os quais a redução no custo de aquisição e operação de *hardware* para executar funções de rede, além de tornar mais flexível o processo de posicionamento e encadeamento dessas funções na infraestrutura [Han et al. 2015].

Embora recente, NFV experimentou diversos avanços em várias frentes, desde o projeto e implantação de funções virtuais de rede (*Virtual Network Functions*, VNFs) [Cohen et al. 2015, Bari et al. 2015] até a operação e gerência das mesmas [Open Networking Lab 2015, Zhang et al. 2016]. Apesar do progresso alcançado, muitas oportunidades de pesquisa permanecem. Um dos tópicos que merece destaque é o posicionamento e encadeamento de VNFs. Em resumo, trata-se de como melhor posicionar VNFs em pontos de presença na rede (*Network Points of Presence*, N-PoPs), e encadeá-las de modo que fluxos de dados sejam processados pelas VNFs na ordem especificada em documentos chamados de SFCs (*Service Function Chains*).

Além de especificar quais funções processarão determinados fluxos, e em qual ordem, as SFCs especificam, ainda, requisitos de poder computacional (nos N-PoPs em que cada função será executada), de largura de banda (entre N-PoPs) e de atraso fim a fim, para processar/encaminhar os fluxos em questão. Para materializar o encadeamento, o paradigma de Redes Definidas por Software (*Software Defined Networking*, SDN) [McKeown et al. 2008] pode ser considerado um aliado conveniente, por permitir que VNFs possam ser posicionadas e encadeadas de forma altamente flexível.

Uma limitação importante das soluções que lidam com o posicionamento e encadeamento de VNFs é que as mesmas consideram os custos de operação das VNFs, e os recursos disponíveis nos N-PoPs, como sendo fixos e imutáveis, ao decidir qual a melhor forma de instanciar um conjunto de SFCs na infraestrutura. Em ambientes reais, no entanto, tanto os custos quanto os recursos disponíveis podem mudar dinamicamente, conforme a carga à qual a rede é submetida. Dessa forma, os requisitos de processamento dos fluxos, tal como especificado nas SFCs, podem ser violados em momentos de alta demanda. Algumas soluções visam a superar essa lacuna analisando o comportamento de uma VNF individual (*firewall*, por exemplo) e ativando novas VNFs como resposta ao aumento da carga de fluxos. A busca individualizada por ótimos locais, como no exemplo do *firewall*, pode não levar a um ótimo global no que se refere ao equilíbrio entre oferta e demanda no posicionamento de funções e encadeamento de fluxos. Mais importante, pode levar a desperdício de recursos, pelo não aproveitamento da capacidade de processamento de fluxos ociosa em VNFs/N-PoPs.

Para suprir essa lacuna, neste artigo propõe-se NFV-PEAR, um *framework* para a orquestração adaptativa de VNFs. O objetivo é permitir o (re)arranjo (periódico) de funções/encadeamentos previamente alocados, em paralelo à instanciação de novas SFCs, visando a lidar com o comportamento dinâmico dos fluxos e flutuações na disponibilidade de recursos nos N-PoPs. Para isso, busca-se tanto (re)encadear fluxos através de VNFs com capacidade de processamento e de larguras de banda ociosas, bem como (re)organizar VNFs entre N-PoPs com maior quantidade de recursos disponíveis. As contribuições do artigo se desdobram em três: (*i*) um modelo formal para assegurar o melhor aprovisionamento de SFCs frente a alterações dinâmicas de demanda e/ou custos

associados aos equipamentos de rede (virtuais ou não); (*ii*) uma arquitetura de referência para o replanejamento e implantação de SFCs, agnóstica de tecnologias de virtualização e infraestrutura; e (*iii*) um mecanismo para aferir dados sobre a operação de VNFs. A partir de resultados obtidos com o NFV-PEAR via avaliação experimental, observou-se um melhor aproveitamento no uso de recursos da rede, ao mesmo tempo que tornou-se possível lidar mais adequadamente com fluxos se comportando de forma dinâmica, sem violar seus requisitos de processamento (tais como determinados nas SFCs).

O restante do artigo está organizado como segue. Na Seção 2 discute-se, a título de motivação, como o desempenho de funções virtualizadas de rede é fortemente influenciado por demandas distintas de tráfego de rede. Na Seção 3 apresenta-se um modelo de programação linear inteira para provisionamento adaptativo de VNFs. Na Seção 4 descreve-se o NFV-PEAR, a solução proposta para a orquestração adaptativa de VNFs. Na Seção 5 discorre-se sobre o ambiente utilizado para avaliação e os principais resultados alcançados. Na Seção 6 discute-se os principais trabalhos relacionados. Por fim, na Seção 7 são apresentadas considerações finais e perspectivas de trabalhos futuros.

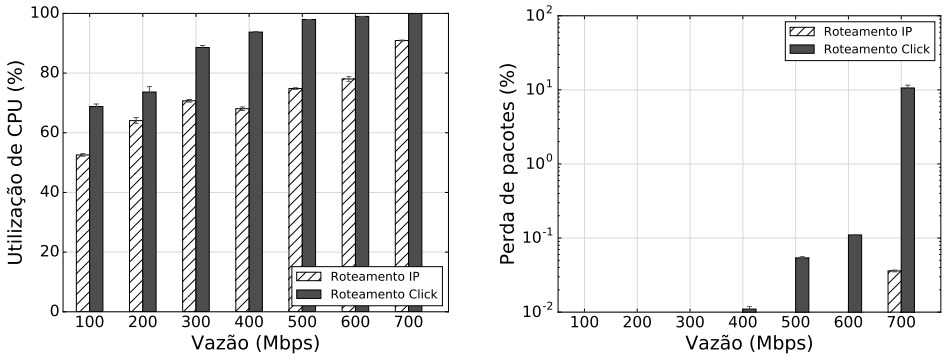
2. O Impacto de Trafego de Rede no Desempenho de Funções de Rede Virtualizadas

No contexto de um arcabouço adaptativo como NFV-PEAR, é fundamental entender como funções virtualizadas de rede estão desempenhando. Como forma de motivar e ilustrar como indicadores de desempenho são afetados à medida que funções de rede são submetidas a diferentes padrões de tráfego, conduziu-se uma série de experimentos em um ambiente NFV típico. A seguir é apresentado um resumo das principais constatações, considerando as métricas de CPU, vazão de dados e perda de pacotes.

Os experimentos foram realizados em ambiente controlado formado por dois servidores A e B, equipados com 1 processador Intel Xeon E5-2420 (1.9GHz, 12 Threads e 15MB cache), 32GB de memória RAM (1333MHz), 1 HD SAS (1TB de capacidade), 1 interface de rede Gigabit e Fedora 21 (kernel 3.17). Os servidores foram conectados diretamente entre si utilizando um cabo de rede Gigabit. No servidor A, foi instalado um *hypervisor* KVM e um *switch* virtual Open vSwitch. No KVM foi instanciada uma máquina virtual com duas interfaces *ethernet* lógicas, 1 vCPU e 1GB de memória RAM. O *switch* virtual foi conectado à interface *ethernet* física e às interfaces da máquina virtual. No servidor B, foram instalados dois *containers* do tipo Docker, cada qual com uma interface *ethernet* lógica, e um *switch* Open vSwitch conectado aos *containers* e à interface *ethernet* física.

O cenário de experimentação foi configurado como segue. No servidor B, os *containers* funcionaram como cliente e servidor Iperf, configurados no modo UDP; no servidor A, a máquina virtual KVM encaminhava os pacotes entre suas interfaces. Durante o experimento, o tráfego iniciava no cliente Iperf, em B, passava pela máquina virtual, em A, e retornava ao servidor Iperf, em B. Optou-se por essa organização para que o custo de geração de tráfego não interferisse no desempenho da máquina virtual, alvo da medição. Ademais, executou-se dois experimentos: (*i*) máquina virtual com roteamento por tabelas de rotas e (*ii*) máquina virtual com roteamento por função de rede executando em Click Router [Kohler et al. 2000]. Os resultados apresentados representam a média de 30 repetições dos experimentos.

A partir dos resultados obtidos, apresentados na Figura 1, percebe-se que conforme a utilização de CPU se aproxima de 100%, perdas de pacote começam a ocorrer, tornando possível prever quando o desempenho da função de rede tende a degradar.



(a) Uso de CPU por vazão. (b) Perda de pacotes por vazão.

Figura 1. Métricas com impacto no desempenho das SFCs.

Nesses mesmos experimentos não foram observadas variações estatisticamente significativas para medições de ocupação de memória. Os resultados observados reforçam a importância do mecanismo adaptativo sendo proposto neste trabalho. NFV-PEAR permitirá o ajuste fino no provisionamento de encadeamentos de funções virtualizadas frente a degradações do desempenho de VNFs ou mudanças no perfil de tráfego.

3. Modelo Formal para Provisionamento Adaptativo de VNFs

Para lidar com o comportamento dinâmico dos fluxos de rede e reorganizar a alocação de VNFs sem desperdício de recursos físicos ou degradação de desempenho, faz-se necessário revisitar os modelos e heurísticas de alocação existentes na literatura e usados para esse fim. Na primeira iteração sobre esse problema, adotou-se uma versão adaptada do modelo proposto por Luizelli et al. [Luizelli et al. 2015, Luizelli et al. 2017], o qual formula o posicionamento e encadeamento estático de funções virtualizadas usando um conjunto de restrições em um sistema linear.

A seguir é descrita a abordagem proposta para provisionamento adaptativo de VNFs, começando pela notação formal (Subseção 3.1) e seguida pelo modelo baseado em Programação Linear Inteira (Subseção 3.2). No modelo, letras superescritas ^{P} e ^{S} indicam símbolos relacionados aos recursos físicos e às SFCs, respectivamente. De forma análoga, as letras superescritas ^{N} e ^{L} referem-se aos N-PoPs/*endpoints*[1] e enlaces que os conectam. Por fim, utiliza-se a letra superescrita ^{H} para denotar subgrafos de uma SFC.

3.1. Notação e Descrição do Modelo

Informações de Entrada. O modelo proposto por Luizelli et al. [Luizelli et al. 2015] considera como entrada um conjunto de SFCs Q e uma infraestrutura física p, esta última sendo uma tripla $p = (N^P, L^P, S^P)$. N^P representa o conjunto de nós da infraestrutura física (N-PoPs ou dispositivos de encaminhamento), enquanto que os pares $(i, j) \in L^P$ representam enlaces físicos unidirecionais. Representa-se enlaces bidirecionais por meio de dois enlaces em direções opostas (isto é, (i, j) e (j, i)). O conjunto de tuplas $S^P = \{\langle i, r\rangle \,|\, i \in N^P \wedge r \in \mathbb{N}^*\}$ contém a localização (representada como um identificador único) de cada N-PoP. O modelo proposto captura as seguintes restrições relacionadas aos recursos físicos: poder computacional dos N-PoPs (representado por c_i^P), bem como largura de banda e *delay* dos enlaces físicos representados, respectivamente, por $b_{i,j}^P$ e $d_{i,j}^P$.

[1]Um *endpoint* é uma entidade que representa, em uma SFC, a origem/destino dos fluxos de dados.

SFCs $q \in Q$ representam qualquer topologia de encaminhamento. Uma SFC é representada por uma tripla $q = (N_q^S, L_q^S, S_q^S)$. O conjunto N_q^S representa os nós virtuais (isto é, *endpoints* e VNFs), enquanto que o conjunto L_q^S representa os enlaces virtuais que os conectam. Note que cada SFC q apresenta, no mínimo, dois *endpoints*, os quais representam regiões específicas da infraestrutura. Os *endpoints* são conhecidos previamente e dados por $S_q^S = \{\langle i, r\rangle \,|\, i \in N_q^S \wedge r \in \mathbb{N}^*\}$. Além disso, cada SFC captura os seguintes requisitos relacionado aos recursos virtuais: processamento requerido por uma VNF i (representado por $c_{q,i}^S$), largura de banda mínima requerida para o tráfego entre VNFs (ou *endpoints*) i e j (representada por $b_{q,i,j}^S$), e latência máxima tolerável entre qualquer par de *endpoints* (representada por d_q^S).

Por simplicidade, assume-se que cada SFC q apresenta um conjunto de caminhos virtuais representado por H_q. Cada elemento $H_{q,i} \in H_q$ representa um caminho possível no subgrafo q, contendo uma origem e um destino. O subconjuntos $N_{q,i}^H \subseteq N_q^S$ and $L_{q,i}^H \subseteq L_q^S$ contém, respectivamente, as VNFs e os enlaces virtuais pertencentes ao caminho $H_{q,i}$.

O conjunto F denota os tipos de VNFs disponíveis (*firewall*, *proxy*, etc). VNFs podem sem instanciadas no máximo U_m vezes. Define-se a função $f_{type} : N^P \cup N^S \rightarrow F$ para o tipo de uma dada VNF, a qual pode estar instanciada em um N-PoP ou ser parte de uma requisição de SFC. Adicionalmente, as funções $f_{cpu} : (F \times U_m) \rightarrow \mathbb{R}_+$ e $f_{delay} : F \rightarrow \mathbb{R}_+$ denotam poder computacional e atrasos relacionadas a uma função virtualizada de rede. Assume-se que as VNFs aprovisionadas podem atender uma demanda maior que a capacidade pré-dimensionada (*overcommitment*). O parâmetro λ $\{\lambda| \in \mathbb{R}_+, \lambda \geq 0\}$ define o percentual da capacidade das VNFs que pode ser violado.

Informações de Saída. A solução do modelo é expressa por conjuntos de variáveis binárias, descritos a seguir. Variáveis $Y = \{\, y_{i,m,j}\,, \forall\, i \in N^P, m \in F, j \in U_m \,\}$ indicam o posicionamento de uma VNF. Isto é, se a instância j da função de rede m está mapeada no N-PoP i. De maneira similar, as variáveis $\overline{Y} = \{\, \overline{y_{i,m,j}}\,, \forall\, i \in N^P, m \in F, j \in U_m \,\}$ indicam que o posicionamento atual de uma VNF j não foi alterado em relação a um dado posicionamento anterior representado por $P_{i,m,j}^y$.

As variáveis $A^N = \{\, a_{i,q,j}^N\,, \forall\, i \in N^P, q \in Q, j \in N_q^S \,\}$ representam a atribuição de uma VNF requisitada (ou um fluxo) a uma VNF aprovisionada. Isto é, a variável indica se o nó j (sendo uma VNF ou um *endpoint*), requerido pela SFC q, é atribuído ao nó i (N-PoP). Similarmente, as variáveis $\overline{A^N} = \{\, \overline{a_{i,q,j}^N}\,, \forall\, i \in N^P, q \in Q, j \in N_q^S \,\}$ indicam que a VNF j (ou o fluxo) da SFC q permanece alocada para uma mesma instância em relação a uma atribuição anterior denotada por $P_{i,q,j}^{a^N}$.

Por último, as variáveis $A^L = \{\, a_{i,j,q,k,l}^L\,, \forall\,(i,j) \in L^P, q \in Q, (k,l) \in L_q^S \,\}$ indicam o aprovisionamento do encadeamento na infraestrutura física, *i.e.* se o enlace virtual (k,l) da SFC q está alocado no enlace físico (i,j). Respectivamente, as variáveis $\overline{A^L} = \{\, \overline{a_{i,j,q,k,l}^L}\,, \forall\,(i,j) \in L^P, q \in Q, (k,l) \in L_q^S \,\}$ indicam que os encadeamentos do enlace virtual (k,l) da SFC q permanece utilizando o enlace físico (i,j).

3.2. Formulação do Modelo

O modelo proposto considera uma função multi-objetivo, a qual minimiza simultaneamente (*i*) os recursos consumidos na infraestrutura (*i.e.*, capacidade de processamento nos N-PoPs, nas VNFs e nos enlaces físicos), e (*ii*) as (possíveis) alterações nos mapeamentos decorrentes da flutuação da demanda alocada (*e.g.*, provisionamento de novas VNFs, readequação nos encadeamentos de SFCs e reatribuição de fluxos às VNFs existentes).

A primeira parte da função objetivo minimiza o consumo de recursos na rede. Essa minimização se materializa pela redução do número de VNFs alocadas (descritas pelas variáveis y) e do comprimento dos encadeamentos realizados (descritos pelas variáveis a^L). Por sua vez, a segunda parte da equação refere-se às alterações realizadas na infraestrutura e está definida por três componentes. O primeiro refere-se à minimização de alterações no posicionamento de VNFs já alocadas (descritas pelas variáveis $\overline{y}$); o segundo refere-se à minimização nas modificações dos encadeamentos existentes (descritas pelas variáveis $\overline{a^L}$); e o terceiro captura as modificações relacionadas com as (re)atribuições de fluxos (ou SFCs) às VNFs (descritas pelas variáveis $\overline{a^N}$). Cada componente é ponderado, respectivamente, por α, β e γ de acordo com as prioridades estabelecidas.

Objetivo:

$$\textbf{Min.} \quad \Big(\sum_{i \in N^P} \sum_{m \in F} \sum_{j \in U_m} y_{i,m,j} + \sum_{(i,j) \in L^P} \sum_{q \in Q} \sum_{(k,l) \in L^S_q} a^L_{i,j,q,k,l} \Big) \quad +$$

$$\Big(- \alpha \cdot \sum_{i \in N^P} \sum_{m \in F} \sum_{j \in U_m} \overline{y_{i,m,j}} - \beta \cdot \sum_{(i,j) \in L^P} \sum_{q \in Q} \sum_{(k,l) \in L^S_q} \overline{a^L_{i,j,q,k,l}} - \gamma \cdot \sum_{i \in N^P} \sum_{q \in Q} \sum_{k \in N^S_q} \overline{a^N_{i,q,k}} \Big)$$

Sujeito a:

$$\sum_{m \in F} \sum_{j \in U_m} y_{i,m,j} \cdot F^{cpu}_{m,j} \leq C^P_i \qquad (\forall i \in N^P) \quad (1)$$

$$\sum_{q \in Q} \sum_{j \in N^S_q : f(j) = f(m)} C^S_{q,j} \cdot a^N_{i,q,j} \leq \lambda \cdot \sum_{j \in U_m} y_{i,m,j} \cdot F^{cpu}_{m,j} \qquad (\forall i \in N^P)(\forall m \in F) \quad (2)$$

$$\sum_{q \in Q} \sum_{(k,l) \in L^S_q} B^S_{q,k,l} \cdot a^L_{i,j,q,k,l} \leq B^P_{i,j} \qquad (\forall (i,j) \in L^P) \quad (3)$$

$$\sum_{i \in N^P} a^N_{i,q,j} = 1 \qquad (\forall q \in Q)(\forall k \in N^S_q) \quad (4)$$

$$a^N_{i,q,k} \cdot l = a^N_{i,q,k} \cdot j \qquad (\forall \langle i,j \rangle \in S^P)(\forall q \in Q)(\forall \langle k,l \rangle \in S^S_q) \quad (5)$$

$$a^N_{i,q,k} \leq \sum_{m \in F} \sum_{j \in U_m : m = f(k)} y_{i,m,j} \qquad (\forall i \in N^P)(\forall q \in Q)(\forall k \in N^S_q) \quad (6)$$

$$\sum_{j \in N^P} a^L_{i,j,q,k,l} - \sum_{j \in N^P} a^L_{j,i,q,k,l} = a^N_{i,q,k} - a^N_{i,q,l} \qquad (\forall q \in Q)(\forall i \in N^P)(\forall (k,l) \in L^S_q) \quad (7)$$

$$\sum_{(i,j) \in L^P} \sum_{(k,l) \in L^H_{q,t}} a^L_{i,j,q,k,l} \cdot D^P_{i,j} + \sum_{i \in N^P} \sum_{k \in N^H_{q,t}} a^N_{i,q,j} \cdot F^{delay}_k \leq D^S_q \qquad (\forall q \in Q)(\forall (N^H_{q,t}, L^H_{q,t}) \in H_q) \quad (8)$$

$$\overline{y_{i,m,j}} = P^y_{i,m,j} \cdot y_{i,m,j} \qquad (\forall i \in N^P)(\forall m \in F)(\forall j \in U_m) \quad (9)$$

$$\overline{a^N_{i,q,j}} = P^{a^N}_{i,q,j} \cdot a^N_{i,q,j} \qquad (\forall i \in N^P)(\forall q \in Q)(\forall k \in N^S_q) \quad (10)$$

$$\overline{a^L_{i,j,q,k,l}} = P^{a^L}_{i,j,q,k,l} \cdot a^L_{i,j,q,k,l} \qquad (\forall (i,j) \in L^P)(\forall q \in Q)(\forall (k,l) \in L^S_q) \quad (11)$$

A seguir descreve-se os conjuntos de restrições que compõem o modelo. Os três primeiros referem-se às limitações de recursos da infraestrutura física. O conjunto de restrições (1) garante que o somatório de todas as instâncias de VNFs aprovisionadas em um dado N-PoP não exceda a capacidade computacional disponível. O conjunto (2) garante que a demanda requerida pelos fluxos das SFCs não exceda a capacidade de processamento aprovisionada para as VNFs. Note que a capacidade aprovisionada das VNFs por ser ultrapassada (em momentos de alta demanda, por exemplo) por um fator λ. Por último, o conjunto (3) garante que as demandas dos encadeamentos aprovisionados em um dado enlace físico não exceda a largura de banda disponível no enlace.

O conjunto de restrições (4)-(6) garante o posicionamento dos recursos virtuais. O conjunto de restrições (4) garante que cada elemento de uma SFC seja mapeado na infraestrutura. Por sua vez, o conjunto (5) garante que os *endpoints* das SFCs sejam mapeados nos dispositivos de rede localizados em regiões específicas da infraestrutura física. O conjunto (6) garante a disponibilidade de instâncias de VNFs nos N-PoPs em que as requisições das SFCs são mapeadas. Isto é, caso uma VNF requisitada por uma SFC seja mapeada em um dado N-PoP i, então (no mínimo) uma instância da VNF estará posicionada e executando em i.

As restrições referentes ao encadeamento das SFCs são descritas pelos conjuntos de restrições (7) e (8). O conjunto (7) garante que haja uma caminho válido na infraestrutura física entre todos os *endpoints* e VNFs da SFC. Por sua vez, o conjunto (8) garante

que os caminhos adotados para encaminhar o tráfego respeite os limites de atraso máximo entre os *endpoints*. A primeira parte da equação refere-se ao atraso proveniente dos enlaces físicos, enquanto que a segunda parte refere-se ao atraso oriundo do processamento de pacotes nas próprias VNFs.

Por fim, os conjuntos de restrições (9)-(11) determinam a similaridade do posicionamento e do encadeamento de SFCs em relação a um dado mapeamento anterior conhecido (denotado pelo conjunto P). Os conjuntos (9), (10) e (11) definem, respectivamente, a similaridade das variáveis relacionadas ao posicionamento de VNFs (variáveis y), relacionadas à atribuição de requisições às VNFs posicionadas (variáveis a^N) e em relação aos encadeamentos adotados (variáveis a^L). Note que o objetivo de tais equações é identificar os casos em que as variáveis da alocação invertem os valores assumidos de 1 para 0. Esses casos, em particular, identificam quando as alocações são modificadas.

4. Posicionamento e Encadeamento Adaptativo de Funções Virtuais de Rede com NFV-PEAR

Tendo sido apresentado o modelo ILP para posicionamento e encadeamento adaptativo de VNFs, nesta seção introduz-se NFV-PEAR: uma proposta de arquitetura para implantação e orquestração de funções de rede. NFV-PEAR apoia-se no modelo ILP proposto para permitir a realocação dinâmica de funções de rede, em resposta a oscilações nas demandas de processamento de fluxos. É importante destacar que a arquitetura foi projetada em sintonia com os principais blocos construtores preconizados pela interface MANO (*Management and Orchestration*) do padrão ETSI [ETSI 2016].

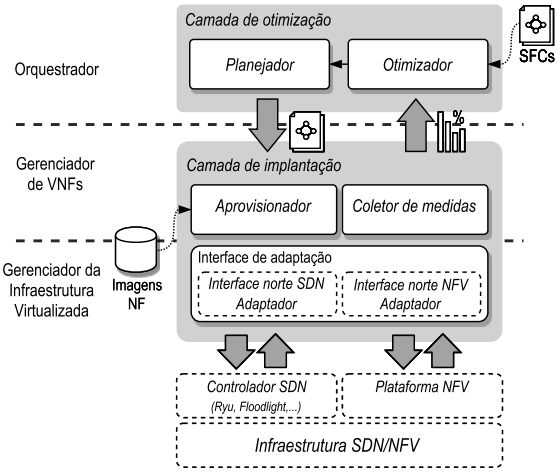


Figura 2. Visão geral da arquitetura proposta.

Uma visão geral da arquitetura proposta é ilustrada na Figura 2, destacando as camadas de *otimização* e de *implantação*. Elas são descritas em detalhes nas Subseções 4.1 e 4.2, respectivamente. A Figura 2 destaca ainda as interações com os controladores SDN e plataforma NFV em uso na infraestrutura, e a relação entre os elementos da arquitetura e os blocos construtores da interface MANO. A camada de otimização da arquitetura proposta, por exemplo, provê os serviços esperados para o bloco construtor "orquestrador" da interface MANO.

Outra peça-chave de NFV-PEAR são as medidas de desempenho, que permitem aferir o estado das VNFs em operação na infraestrutura e identificar realocações necessárias para responder a flutuações no volume de fluxos. As métricas propostas no

escopo deste artigo, e a metodologia empregada para aferir a importância das mesmas, são descritas na Seção 2.

4.1. Camada de Otimização

A camada de otimização reúne os módulos responsáveis pela otimização e pelo planejamento da instanciação e do encadeamento de SFCs na infraestrutura. Observe nesse caso que tanto SFCs já implantadas como as em implantação são processadas por esses módulos, ao se (re)planejar a alocação de VNFs contidas nas mesmas.

O módulo *otimizador* é responsável por computar a melhor alocação possível de VNFs na rede, considerando o conjunto de SFCs mencionado anteriormente, bem como informações sobre o estado atual da infraestrutura (*endpoints*, N-PoPs/VNFs e recursos disponíveis nos mesmos, enlaces, etc.). Para esse fim, o otimizador implementa o modelo ILP discutido na seção anterior. A saída desse módulo – a solução para o modelo ILP no cenário dado – é repassada ao planejador.

O módulo *planejador* é responsável por determinar, algoritmicamente, a melhor forma de conduzir, na prática, as alterações necessárias na alocação de VNFs na rede e nos encadeamentos correspondentes. Seu objetivo é manter a infraestrutura em um estado próximo ao de operação ótima, efetuando, para isso, o mínimo de mudanças necessárias. Diversas estratégias podem ser adotadas para assegurar transições suaves entre estados que a infraestrutura deve assumir e, com isso, evitar disrupções. Tais não fazem parte do escopo deste artigo e serão detalhadas em um trabalho futuro.

4.2. Camada de Implantação

A camada de implantação reúne os módulos responsáveis por provisionar as SFCs na rede física. O *aprovisionador* é responsável por implementar na rede as alocações de VNFs e os encadeamentos, conforme os mapeamentos de SFCs recebidos da camada de otimização. O módulo *coletor de medidas* implementa as funcionalidades de monitoração das VNFs implantadas na rede, consolidando estatísticas de operação das mesmas, as quais são repassadas para a camada de otimização.

Ambos os módulos comunicam-se com a *interface de adaptação* para realizar as atividades de orquestração/monitoração de VNFs na infraestrutura física. Essa interface é composta por dois sub-módulos, (*i*) *interface norte SDN*, responsável por traduzir requisições de instalação de encadeamentos e consultas de estado (por exemplo, nos *switches*) para o protocolo utilizado pelo controlador SDN, e (*ii*) *interface norte NFV*, responsável por adaptar requisições pertinentes às VNFs ao protocolo utilizado pelo controlador NFV da infraestrutura.

Para facilitar o processo de integração com outras soluções compatíveis com a interface MANO, os módulos da camada de implantação expõem uma interface de programação (API) para orquestração e implantação simplificada de VNFs. A API é projetada de modo a reduzir a complexidade de programação da rede SDN e facilitar o gerenciamento das funções virtuais. Em resumo, a API expõe métodos para instanciação de VNFs, posicionamento de VNFs nos N-PoPs e instanciação de encadeamentos.

5. Avaliação

Para aferir a eficácia e a efetividade de NFV-PEAR, definiu-se um processo sistemático de avaliação. Para tal, o modelo formalizado na Seção 3 foi implementado e executado no *CPLEX Optimization Studio* versão 12.4[2]. Os experimentos foram conduzidos em uma

[2]http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/

máquina com quatro processadores Intel i5 2.6 GHz, 8 GB de memória RAM, executando sistema operacional Ubuntu/Linux Server 11.10 x86_64.

5.1. Carga de Trabalho

Para realizar os experimentos, adotou-se uma estratégia similar à empregada em trabalhos anteriores na área [Luizelli et al. 2015]. A infraestrutura física foi gerada com a ferramenta Brite[3] utilizando o modelo Barabasi-Albert (BA-2) [Albert and Barabási 2000]. O modelo adotado apresenta características topológicas semelhantes às infraestruturas típicas de ISPs (*Internet Service Providers*). As infraestruturas físicas consideradas contêm um total de 50 N-PoPs, cada um com capacidade computacional de 100%. Em média, cada rede apresenta 300 enlaces com capacidade uniforme de 10 Gbps e atrasos médios de 10ms. Os N-PoPs são posicionados em locais diversos na rede.

Considera-se dois tipos de imagens de VNFs disponíveis para instanciação. Para cada tipo de VNF, assume-se a disponibilidade de instâncias de pequena e grande capacidade computacional (demandando, respectivamente, 25% e 100% da capacidade computacional de um N-PoP). Para a avaliação conduzida, considera-se um conjunto de 20 SFCs sendo submetidas à infraestrutura. Os tipos das VNFs requisitadas pelas SFCs são aleatoriamente selecionadas dentre as possíveis. Cada VNF requisita entre 25% e 50% da capacidade de uma imagem instanciada em um N-PoP. As SFCs consideradas seguem uma topologia em linha com os seus *endpoints* selecionados aleatoriamente na infraestrutura física.

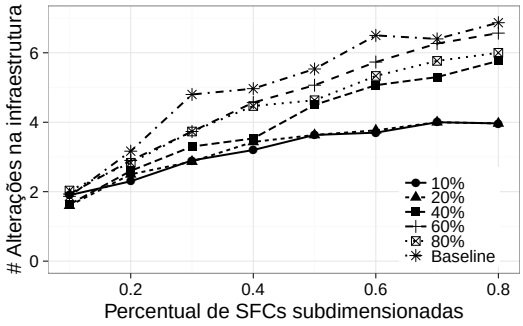
Para avaliar a capacidade do modelo em replanejar a infraestrutura com o mínimo de interrupções, considera-se que um percentual das SFCs aprovisionadas alternam entre um modo de consumo considerado normal (ou pré-aprovisionado) e um modo de consumo acima do planejado (por exemplo, demanda de pico). Nesse último caso, replanejamentos pontuais são necessários para manter o desempenho e a estabilidade do sistema. Compara-se o modelo proposto com o modelo de Luizelli et al. [Luizelli et al. 2015]. Naquele, quando há a necessidade de replanejamento, todas as SFCs são ressubmetidas e aprovisionadas na infraestrutura.

5.2. Resultados

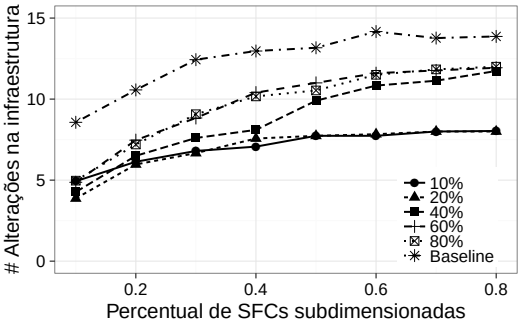
A avaliação concentra-se principalmente na qualidade das soluções geradas pelo otimizador. Primeiramente, avalia-se o número de modificações necessárias na infraestrutura em função da variação na demanda. A Figura 3 apresenta a quantidade média de modificações relacionadas com: (*i*) o reposicionamento de VNFs (Figura 3(a)), (*ii*) a reatribuição de SFCs às VNFs (Figura 3(b)) e (*iii*) o reencadeamento de SFCs (Figura 3(c)). Varia-se a proporção de SFCs subdimensionadas (*i.e.*, aquelas que apresentam demandas maiores que o aprovisionado) de 10% a 80% do total de SFCs alocadas na infraestrutura. Além disso, varia-se o percentual da demanda excedida de 10% a 80% (curvas distintas). Para esses experimentos, considera-se os valores de α, β, γ e λ (do modelo ILP) iguais a 1.

Observa-se que o número de alterações necessárias (eixo y) para readequar a rede à nova demanda é proporcional 1) ao percentual de SFCs com aumento de demanda e 2) aos valores de demanda excedidos (eixo x). Além disso, observa-se que o número de alterações relacionadas ao reposicionamento de VNFs é substancialmente menor em comparação ao observado para reatribuições de fluxos e reencadeamentos de SFCs. Isso indica a factibilidade do modelo proposto em ambientes reais, uma vez que o tempo requerido para instanciar (ou migrar) uma VNF é substancialmente maior (na ordem de

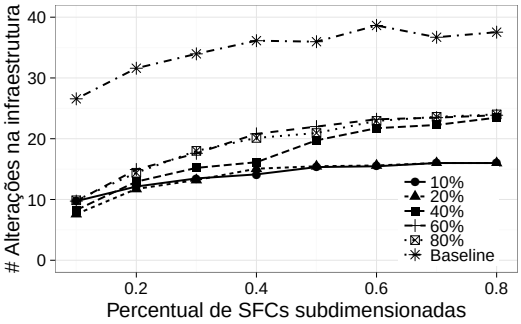
[3]http://www.cs.bu.edu/brite/



(a) Alterações no posicionamento de VNFs.



(b) Alterações nas atribuições de SFCs às VNFs.



(c) Alterações nos encadeamentos das SFCs.

Figura 3. Análise do replanejamento da alocação de SFCs.

milissegundos a segundos) que o de reprogramar um dispositivo de encaminhamento (na ordem de milissegundos), por exemplo. Ainda, em comparação com o *baseline*, observa-se que o modelo proposto reduz em 25% o número de alterações relacionadas ao posicionamento de VNFs e em até duas vezes a quantidade de alterações relacionadas ao encadeamento e reatribuições de SFCs nas VNFs.

A seguir, avalia-se o impacto do fator de *overcommitment* (parâmetro λ) de VNFs no replanejamento da infraestrutura. Note que quanto maior é o fator de *overcommitment*, maior é a chance de uma determinada VNF apresentar degradação de desempenho. Para esta avaliação, fixou-se o percentual de SFCs subdimensionadas em 80% e o percentual de aumento da demanda em 40%. O parâmetro λ é variado entre 0 e 40%. A Figura 4 ilustra o número de alterações necessárias para readequar a demanda de tráfego para esse cenário. Observa-se que quanto maior é o fator de *overcommitment*, menor é o número de alterações necessárias na infraestrutura. Por exemplo, com 10% de *overcommitment*, há uma de redução de 30% no número total de alterações.

Por fim, discute-se o tempo médio necessário para encontrar uma solução ótima para o problema de replanejamento do encadeamento de SFCs. Em todos os experimentos, o tempo médio necessário para a resolução do modelo proposto permanece abaixo de 2 segundos. Apesar do problema apresentado e modelado ser NP-Difícil, tais resultados indicam que o modelo exato pode ser aplicado para instâncias do problema de pequena e média escala. Para ser aplicada em infraestruturas com escalas maiores, avaliações adicionais são necessárias para identificar os limites de tempo de computação.

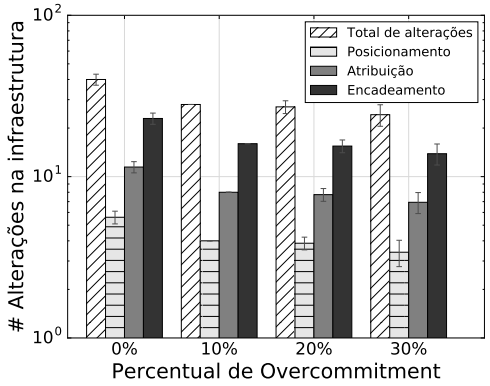


Figura 4. Impacto de *overcommitment* no replanejamento da infraestrutura.

6. Trabalhos Relacionados

Os trabalhos de pesquisa no âmbito de NFV podem ser organizados considerando várias perspectivas. Na área de orquestração, um dos esforços mais notórios é o *Open Platform for NFV* (OPNFV) [Open Networking Lab 2015]. Essa plataforma *open source* visa a fomentar a interoperabilidade entre as tecnologias habilitadoras de NFV (por exemplo, Open vSwitch, KVM e Xen) com as demais camadas da arquitetura proposta pelo ETSI [ETSI 2016] (por exemplo, de orquestração e de monitoramento).

Em paralelo, várias outras plataformas tem sido propostas para superar lacunas específicas na orquestração de VNFs, por exemplo o ClickOS [Martins et al. 2014], o OpenNetVM [Zhang et al. 2016] e o VirthPhy [Dominicini et al. 2016]. O ClickOS, baseado no hipervisor Xen e usando funções virtuais escritas em Click [Kohler et al. 2000], visa a reduzir a sobrecarga de cópia de pacotes entre interfaces, permitindo alcançar vazão próxima da velocidade do enlace. O OpenNetVM, por sua vez, introduz uma camada de encaminhamento virtual para integrar o mecanismo de virtualização leve Docker à biblioteca de aceleração de pacotes Intel DPDK. Por fim, VirtPhy apresenta uma plataforma programável para *data centers* de pequeno porte, nos quais tanto as funções quanto os elementos de rede que as interconectam são virtualizados.

Na área de monitoração, uma das principais iniciativas é o NFV-VITAL [Cao et al. 2015]. Nesse trabalho, os autores propõem um *framework* para caracterizar o desempenho de VNFs executando em ambientes de nuvem. A partir dessa caracterização, torna-se possível (*i*) estimar a melhor alocação de recursos computacionais para executar as VNFs, e (*ii*) determinar o impacto de diferentes configurações de virtualização e de *hardware* no desempenho das VNFs. Outra iniciativa é o NFVPerf [Naik et al. 2016], uma ferramenta para detecção de gargalos em ambientes NFV. Por meio da análise dos fluxos de dados que transitam entre as VNFs, torna possível calcular vazão e atrasos médios, e assim possíveis degradações de desempenho em tempo real.

No âmbito de posicionamento e encadeamento de VNFs, vários trabalhos merecem destaque, entre os quais os de Bari et al. [Bari et al. 2015] e Luizelli et al. [Luizelli et al. 2015, Luizelli et al. 2017]. Bari et al. [Bari et al. 2015] descrevem o problema de orquestração de VNFs, que consiste em determinar o número de VNFs e suas localizações na rede de modo que os custos operacionais sejam ótimos. Os autores formulam o problema via sistema linear (*Integer Linear Programming*, ILP), e utilizam CPLEX e programação dinâmica para otimizar as alocações em ambientes NFV de menor escala. Mais recentemente, Luizelli et al. [Luizelli et al. 2017] abordaram o problema

para ambientes de larga escala, via proposta de um algoritmo de otimização que combinam programação matemática e meta-heurísticas de busca.

Apesar dos avanços observados, as soluções existentes não abordam cenários de flutuações e gargalos localizados que ocorrem devido a variações no volume de fluxos em trânsito na rede. Uma estratégia *ad hoc* para lidar com essas flutuações é reexecutar os algoritmos de alocação de VNFs, e reorganizá-las conforme o resultado obtido. Apesar de efetiva, essa estratégia é computacionalmente mais cara (por reexecutar globalmente os algoritmos de otimização), e não permite reagir eficientemente à dinamicidade no comportamento dos fluxos. O trabalho de Rankothge et al. [Rankothge et al. 2015] é o que mais se aproxima de uma solução efetiva para esse problema. Nele, os autores utilizam algoritmos genéticos para introduzir funções de rede com capacidade de processamento escalável. No entanto, as funções de rede são consideradas de forma isolada, portanto sem levar em conta possíveis otimizações globais como, por exemplo, reencadear fluxos com requisitos similares para VNFs de maior capacidade.

Considerando as limitações discutidas acima, NFV-PEAR apresenta-se como uma solução para readequar a rede frente as variações de demanda, via identificação de gargalos no processamento de fluxos, reorganização do posicionamento e encadeamento de funções de rede localmente/globalmente, e visando à minimização da disrupção no processamento dos fluxos em trânsito.

7. Considerações Finais

Neste trabalho propôs-se NFV-PEAR, um *framework* para orquestração adaptativa de funções de rede em ambientes NFV. As contribuições deste trabalho se desdobram em (*i*) um modelo formal para assegurar o melhor aprovisionamento de SFCs frente a alterações dinâmicas de demanda e/ou custos associados aos equipamentos de rede, (*ii*) uma arquitetura de referência para o replanejamento e implantação de SFCs, agnóstica de tecnologias de virtualização e infraestrutura e (*iii*) um conjunto de métricas para representar dados sobre a operação de VNFs. Após a formalização do modelo ótimo para o problema de replanejamento adaptativo do encadeamentos de funções virtualizadas de rede, realizou-se uma avaliação analítica. Os resultados obtidos evidenciaram que o modelo proposto contribui significativamente para uma redução no número de modificações na infraestrutura física (de até 25% no reposicionamento de VNFs e de mais de 200% no reencadeamento de encadeamentos de funções de rede).

Como perspectivas de trabalhos futuros, pretende-se materializar o modelo e a arquitetura propostos em um ambiente de orquestração real. O objetivo é avaliar métricas de desempenho como tempo de resposta e de reação do sistema. Outra proposta de trabalho futuro é estender a avaliação para identificar correlações na valoração dos parâmetros do modelo (em especial, α, β e γ) na qualidade das soluções obtidas. Por fim, visa-se a desenvolver e integrar métodos de predição de demanda de tráfego ao modelo proposto.

Referências

Albert, R. and Barabási, A.-L. (2000). Topology of evolving networks: Local events and universality. *Physical Review Letters*, 85:5234 – 5237.

Bari, M. F., Chowdhury, S. R., Ahmed, R., and Boutaba, R. (2015). On orchestrating virtual network functions. In *Proceedings of the 2015 11th International Conference on Network and Service Management (CNSM)*, CNSM '15, pages 50–56.

Cao, L., Sharma, P., Fahmy, S., and Saxena, V. (2015). NFV-VITAL: A framework for characterizing the performance of virtual network functions. In *2015 IEEE Conference*

on Network Function Virtualization and Software Defined Network (NFV-SDN), pages 93–99.

Cohen, R., Lewin-Eytan, L., Naor, J. S., and Raz, D. (2015). Near optimal placement of virtual network functions. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1346–1354.

Dominicini, C. K., Vassoler, G. L., Ribeiro, M. R., and Martinello, M. (2016). Virtphy: A fully programmable infrastructure for efficient nfv in small data centers. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*.

ETSI (2016). Network Functions Virtualisation (NFV). Available: <http://www.etsi.org/technologies-clusters/technologies/nfv>.

Han, B., Gopalakrishnan, V., Ji, L., and Lee, S. (2015). Network function virtualization: Challenges and opportunities for innovations. *Communications Magazine, IEEE*, 53(2):90–97.

Kohler, E., Morris, R., Chen, B., Jannotti, J., and Kaashoek, M. F. (2000). The click modular router. *ACM Transactions on Computer Systems (TOCS)*, 18(3):263–297.

Luizelli, M., Bays, L., Buriol, L., Barcellos, M., and Gaspary, L. (2015). Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 98–106.

Luizelli, M. C., da Costa Cordeiro, W. L., Buriol, L. S., and Gaspary, L. P. (2017). A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining. *Computer Communications*, 102:67 – 77.

Martins, J., Ahmed, M., Raiciu, C., Olteanu, V., Honda, M., Bifulco, R., and Huici, F. (2014). Clickos and the art of network function virtualization. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 459–473, Seattle, WA. USENIX Association.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Naik, P., Shaw, D. K., and Vutukuru, M. (2016). NFVPerf: Online Performance Monitoring and Bottleneck Detection for NFV. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*.

Open Networking Lab (2015). Open Platform for NFV (OPNFV). Disponível em <https://www.opnfv.org/>. Acesso em: 28 jul. 2016.

Rankothge, W., Le, F., Russo, A., and Lobo, J. (2015). Experimental results on the use of genetic algorithms for scaling virtualized network functions. In *IEEE SFV-SDN*.

Zhang, W., Liu, G., Zhang, W., Shah, N., Lopreiato, P., Todeschi, G., Ramakrishnan, K., and Wood, T. (2016). OpenNetVM: A Platform for High Performance Network Service Chains. In *Proceedings of the 2016 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization. ACM*.

Trilha Principal do SBRC 2017
Sessão Técnica 20
Redes Ópticas

Novo Esquema para Provisão de Modulação Adaptativa em Redes Ópticas Elásticas

Lucas R. Costa, André C. Drummond

Departamento de Ciência da Computação – Universidade de Brasília (UnB),
70910-900, Brasília, Brasil

lucasrc.rodri@gmail.com, andred@unb.br

Abstract. *Elastic optical networks are becoming a promising technology for the future of high-capacity networks. Its features provide flexibility and superior scalability in spectrum allocation following the growing demands of Internet traffic. In this paper we propose a novel approach to address the Routing, Modulation Level, and Spectrum Allocation (RMLSA) problem through the use of a distance-adaptive modulation scheme that enables the routing of traffic through multiple hops in virtual topology, enabling smoothing out the spectrum continuity and transmission distance constraints. The results showed that the use of the adaptive modulation scheme proposed may provide a reduction of up to* 82% *in the bandwidth blocking ratio and energy savings of up to* 34% *compared to literature approaches. This proposal opens a new avenue for future research, allowing new solutions for RMLSA problem.*

Resumo. *Redes ópticas elásticas mostram-se como uma tecnologia promissora para o futuro das redes de alta capacidade. Suas características proporcionam uma flexibilidade e escalabilidade superior na alocação de espectro acompanhando a crescente demanda do tráfego na Internet. Este trabalho tem como objetivo propor uma nova abordagem para o problema de roteamento e atribuição de espectro com modulação adaptativa (Routing, Modulation Level, and Spectrum Allocation - RMLSA) por meio do uso de um esquema de modulação adaptativa que viabilize o roteamento do tráfego através de múltiplos saltos na topologia virtual, permitindo a suavização das restrições de continuidade de espectro e distância de transmissão. Os resultados mostraram que o uso do esquema de modulação adaptativa proposto pode proporcionar uma redução de até* 82% *na taxa de bloqueio de banda e uma eficiência energética de até* 34% *em comparação com abordagens da literatura. Esta proposta abre um novo caminho para pesquisas futuras, permitindo novas soluções para o problema RMLSA.*

1. Introdução

O paradigma de redes ópticas elásticas (Elastic Optical Networks - EON) tem atraído bastante atenção e interesse na academia e na industria nos últimos anos [Tomkos et al. 2014, Chatterjee et al. 2015]. A característica fundamental das redes EON refere-se à capacidade da rede de ajustar dinamicamente seus recursos, tais como, a largura de banda e formato de modulação de acordo com os requisitos de cada demanda [Jinno et al. 2010]. Em comparação com as redes ópticas tradicionais de multiplexação por divisão de comprimento de onda (*Wavelength-Division Multiplexing* - WDM), as redes EON exigem

mecanismos de alocação de largura de banda mais sofisticados, baseados em abordagens de roteamento e de atribuição do espectro (*Routing and Spectrum Assignment* - RSA) e dispositivos de largura de banda variável, como transmissores de largura de banda variável (*Bandwidth Variable Transponder* - BVT) e comutadores ópticos de banda variável (*Bandwidth Variable Optical Cross-Connects* - BV-OXC) [Chatterjee et al. 2015].

A eficiência do espectro esta relacionada à sua tecnologia OFDM (*Orthogonal Frequency Division Multiplexing*) implementada pelos BVTs e BV-OXCs das redes EON. Estes componentes possibilitam que o espectro seja separado em subportadoras contíguas parcialmente sobrepostas. Isso permite a criação de caminhos ópticos com taxa de transmissão ajustável facilitando o atendimento das recentes requisições de tráfego da Internet [Tomkos et al. 2014]. Para melhorar ainda mais a eficiência e a utilização do espectro, a literatura EON incorporou a qualidade de transmissão (*Quality-of-Transmission* - QoT) para as soluções RSA. Isto transformou o problema RSA no problema de alocação de espectro com modulação adaptativa (*Routing, Modulation Level, and Spectrum Allocation* - RMLSA), que adiciona a atribuição do formato de modulação ao espectro, considerando sua distância de transmissão [Christodoulopoulos et al. 2011].

Este trabalho propõe uma técnica que resolve o problema de roteamento e atribuição de espectro com modulação adaptativa (RMLSA) através do uso de um esquema de modulação adaptativa inovador. Este esquema propõe dividir o problema RMLSA em dois subproblemas: (*i*) atribuição do formato de modulação; e (*ii*) execução de um algoritmo RSA através do roteamento *multi-hop*. O esquema proposto permite a utilização de qualquer abordagem RSA clássica para resolver o problema RMLSA ao mesmo tempo que ameniza as restrições de continuidade de espectro e de distância de transmissão do problema. Ao decompor o problema em dois sub-problemas, estabelecem-se novas possibilidades a serem exploradas em futuras pesquisas, propiciando o surgimento de novas soluções RMLSA adaptadas às necessidades dos provedores de redes de transporte óptico.

Uma análise preliminar deste estudo foi apresentado em [Costa and Drummond 2016b], onde os autores abordam as vantagens do esquema em relação a ganhos da taxa bloqueio e eficiência de espectro. Neste trabalho novas perspectivas, como eficiência energética, são observadas. Apresenta-se novos resultados e novas métricas que consolidam ainda mais a relevância do esquema proposto. Os resultados mostram uma redução de até 82% na taxa de bloqueio de banda e uma eficiência energética de até 34% em comparação com abordagens da literatura.

O restante do artigo está organizado como segue. A Seção 2 apresenta brevemente os conceitos básicos da rede EON e o modelo de gasto energético adotado nas avaliações. A Seção 3 apresenta os trabalhos relacionados ao problema RMLSA. A Seção 4 apresenta o esquema de modulação adaptativa proposto para resolver o problema RMLSA. A Seção 5 mostra as avaliações de desempenho do esquema de modulação adaptativa proposto. Por fim a Seção 6 apresenta as considerações finais.

2. Redes Ópticas Elásticas

As redes EON possuem a característica de dividir os recursos espectrais em *slots* de frequência na forma de subportadoras OFDM, permitindo múltiplos formatos de modulação e taxas de dados e espectro de tamanhos variados. Uma rede EON é capaz de alocar uma

demanda em um caminho óptico com largura de banda apropriado à requisição. Como já mencionado, a arquitetura da rede EON é composta por BVTs e BV-OXCs, estes componentes são responsáveis pelo estabelecimento de caminhos ópticos flexíveis. Os BVTs são encarregados por alocar espectro suficiente para acomodar cada demanda. Por outro lado, os BV-OXC são os elementos responsáveis por estabelecer um caminho óptico fim-a-fim com a largura de banda necessária para acomodar os recursos espectrais estabelecidos pelos BVTs [Zhang et al. 2013].

A alocação do espectro em diferentes níveis de modulação é outro aspecto que permite o ajuste flexível da largura de banda em uma rede EON. Em particular, alocação de espectro com modulação adaptativa permite com que cada caminho óptico, possa ser modulado individualmente utilizando um modulador (BVT) diferente para cada transmissão [Nag et al. 2010]. A escolha do nível de modulação deve levar em consideração a qualidade necessária de transmissão (QoT) e, consequentemente, a tolerância de relação sinal-ruído óptico [Wan et al. 2011]. Uma abordagem comumente utilizada pela literatura EON define a distância de transmissão do caminho óptico como o fator mais relevante na QoT [Jinno et al. 2010]. Portanto, a escolha do formato de modulação a ser utilizado é estabelecida de acordo com a distância do caminho percorrido na fibra.

A tecnologia de agregação de tráfego elétrica (EG) das redes ópticas tradicionais também é aplicável nas redes EON. A EG permite a agregação de múltiplas requisições de tráfego em um mesmo caminho óptico através da multiplexação por divisão de tempo [Zhang et al. 2011]. A tecnologia OFDM também permite outro tipo de agregação do tráfego, a agregação através do meio óptico. A agregação óptica (OG) permite aproveitar a flexibilidade proporcionada pelos comutadores EON agregando múltiplos caminhos ópticos em um único transmissor sem a necessidade de bandas de guarda entre os caminhos comutando-os opticamente em conjunto [Zhang et al. 2012]. Esse agrupamento é realizado para suportar agregação de modo transparente, sem conversão do sinal do domínio óptico para elétrico. Esse grupo de caminhos ópticos é chamado de túnel óptico.

O roteamento em múltiplos saltos (*multi-hop*) é dado quando uma conexão passa por vários caminhos ópticos, tendo em vista que cada caminho óptico pode passar por vários enlaces. Ao final de cada caminho óptico, será realizada uma conversão OEO (*Optical-Electrical-Optical*) o que resultará em um salto na topologia virtual. Os caminhos ópticos da conexão podem ser novos ou existentes. Novos quando os caminhos ópticos forem criados para atender uma nova conexão, existentes quando os caminhos ópticos forem usados através da agregação de tráfego.

Em um cenário de rede dinâmico, o processo de estabelecimento e encerramento de conexões inevitavelmente cria pequenos fragmentos de espectro não-contíguos o que conduz ao chamado problema de fragmentação de espectro [Zhang et al. 2013]. Este problema também causa a utilização ineficiente dos recursos de espectro proporcionando uma degradação do desempenho da rede. Para medir o nível de fragmentação do canal pode-se utilizar o cálculo da fragmentação externa, comumente utilizado na arquitetura de computadores, e denotada pela Equação 1 [Horota et al. 2014]:

$$F_{ext} = 1 - \frac{maiorBlocoLivre}{totalLivre} \quad (1)$$

2.1. Modelo de Consumo Energético das redes EON

Devido à indisponibilidade comercial dos componentes de largura de banda variável OFDM (BVT/BV-OXC) algumas hipóteses têm sido feitas pela literatura para mensurar o gasto energético dos componentes EON [Vizcaíno et al. 2012]. O consumo de energia de um único BVT pode ser interpolado em função da sua taxa de transmissão (TR) como mostra a Equação 2. Consequentemente, o consumo energético dos BVTs para uma subportadora nos diferentes formatos de modulação é apresentado na Tabela 1. Além disso, um adicional de 25% é considerado para possíveis sobrecargas nos BVTs [Vizcaíno et al. 2012].

$$PC(W) = 1,25 \times TR(Gb/s) + 31,5 \tag{2}$$

A Tabela 1 também mostra as distâncias máximas necessárias para que as modulações obtenham uma taxa de transmissão com QoT aceitável na rede.

Tabela 1. Consumo Energético dos transmissores de banda variável (BVT) para diferentes formatos de modulação.

Formato de Modulação	Capacidade da Subportadora (Gb/s)	Consumo Energético (W)	Distância Máxima (km)
BPSK	12,5	47,13	8000
QPSK	25	62,75	4000
8QAM	37,5	78,38	2000
16QAM	50	94	1000
32QAM	62,5	109,63	500
64QAM	75	125,23	250

Semelhante à tecnologia de rede tradicional, o consumo energético dos comutadores ópticos de banda variável (BV-OXC) é dado em função do grau de conexão de cada nó N. Além disso, é adicionada uma sobrecarga de $150\ W$ por nó como mostra a Equação 3.

$$PO(W) = N \times 85 + 150 \tag{3}$$

Em relação aos amplificadores ópticos (*Optical Line Amplifiers* - OLA) EDFA (Erbium Doped Fiber Amplifier), são considerados $60\ W$ por par de fibra óptica ($30\ W$ por fibra E). Além disso, é considerado mais $140\ W$ adicionais por OLA como possível sobrecarga, como mostra a Equação 4.

$$PA(W) = E \times 30 + 140 \tag{4}$$

3. Trabalhos Relacionados

Recentemente, há um número crescente de trabalhos de pesquisa que investigam soluções para o problema de roteamento e atribuição de espectro em redes EON. Os problemas de alocação de espectro são investigados sob diversos cenários de rede com roteamento *single-hop* ou *multi-hop*, incluindo ainda considerações sobre a tecnologia de modulação e distância adaptativa do canal.

O problema RMLSA foi inicialmente apresentado em [Christodoulopoulos et al. 2011], onde os autores adicionam o formato de modulação ao problema RSA. Posteriormente, o problema RMLSA foi investigado em [Wan et al. 2012], onde os autores avaliam os efeitos da modulação adaptativa em algoritmos RSA. Os autores propõem um esquema de modulação adaptativa, denominado *mAdap*, que interage através das possíveis modulações da rede, em ordem decrescente na quantidade de bits por símbolo, aplicando um algoritmo RSA até encontrar uma solução. Os resultados de simulação para este trabalho mostraram uma redução significativa no bloqueio e no uso do espectro.

Os autores em [Ye et al. 2014] propõem o primeiro algoritmo de modulação adaptativa com técnicas de agregação de tráfego. No algoritmo proposto, as demandas de tráfego com mesma origem que compartilham o mesmo enlace são agregadas opticamente e comutadas pelo meio óptico sem a utilização de bandas de guarda. Embora o algoritmo leve em consideração o uso da modulação adaptativa, o trabalho não leva em consideração o ambiente de tráfego de rede dinâmico, somente a abordagem estática. Outro fator importante não investigado é o uso de múltiplos saltos na topologia virtual.

Até o momento, são poucos os trabalhos que consideram o uso do roteamento *multi-hop* e o cenário de modulação adaptativa em EON. Em [Costa and Drummond 2016a, Costa et al. 2016b], os autores apresentaram uma abordagem RMLSA *multi-hop* que procura usar técnicas de agregação de tráfego associado ao nível de modulação mais eficiente do ponto de vista espectral. Os resultados foram comparados com várias abordagens RSA utilizando o esquema *mAdap*.

Observando as vantagens do roteamento *multi-hop* com o uso de modulação adaptativa, os autores apresentam em [Costa and Drummond 2016b] uma análise preliminar do estudo proposto neste trabalho. Os resultados têm como foco a diminuição da taxa de bloqueio de banda e a redução do uso de espectro na rede. Conclui-se que é vantajosa a utilização do esquema proposto. Com o intuito de enfatizar mais as vantagens proporcionadas pelo esquema, este trabalho apresenta a avaliação de novas métricas em novos cenários de rede, mostrando que, além da redução na taxa de bloqueio de banda, o esquema ainda proporciona uma eficiência energética na rede.

4. Proposta de Trabalho

A proposta deste trabalho visa criar um esquema de modulação adaptativa, baseado no *mAdap*, para resolver o problema RMLSA. O esquema, denominado AMMS (*Adaptive Modulation Multi-hop Schema*), permite a utilização de qualquer abordagem RSA clássica para resolver o problema RMLSA. O AMMS tem como objetivo atribuir níveis de modulação adequados, associado a uma quantidade de saltos necessários para atender uma requisição de tráfego. Para atingir esse objetivo o AMMS caracteriza a rede em zonas de alcançabilidade.

O AMMS implementa as zonas de alcançabilidade através de gráficos auxiliares que representam o alcance relativo de cada nó para um nível de modulação específica. A partir da topologia física, o AMMS constrói topologias de modulação (off-line) cujas arestas representam a zona de alcançabilidade de cada nó. Cada aresta na determinada topologia de modulação é construída a partir do caminho mais curto entre os nós, satisfazendo o alcance do nível de modulação adotado para fornecer um QoT aceitável na rede.

A Figura 1 mostra as topologias virtuais de modulação para uma topologia física com 7 nós. Neste exemplo, os formatos de modulação BPSK e QPSK foram considerados permitindo alcances máximos de 8000 e 4000 km, respectivamente, de acordo com a Tabela 1. Pode-se observar que em cada topologia as arestas são construídas a partir dos alcances entre os nós. O peso da aresta representa a menor distância entre os nós (a soma das arestas que compõe a menor distância). Portanto, isso representa que cada par de nós ligados por uma aresta pode estabelecer um caminho óptico fim-a-fim no determinado nível de modulação. Por exemplo, considere um pedido de conexão do nó P1 para o nó P4, é possível atender a conexão com apenas um caminho óptico ($B1 \rightarrow B4$) no nível de modulação BPSK, no entanto são necessários no mínimo 2 caminhos ópticos ($Q1 \rightarrow Q2 \rightarrow Q4$) para atender a conexão no nível de modulação QPSK.

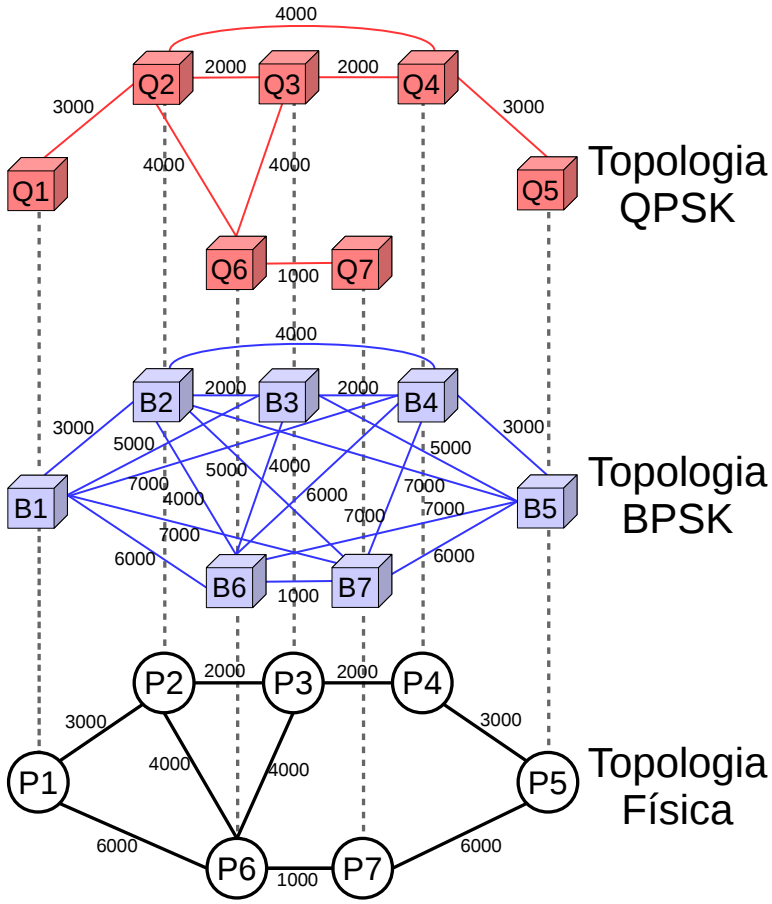


Figura 1. Exemplos de topologias virtuais de modulação para uma topologia com 7 nós. As distâncias dos enlaces são apresentadas em quilômetros.

A ideia geral é dividir a rota da requisição de tráfego em várias partes procurando sempre atender os níveis de modulação mais adequados a essa requisição. Para isso, o esquema é composto por uma estrutura de dados, computada off-line, formada a partir dos $k-$menores caminhos entre cada par de nós de cada topologia virtual de modulação. O AMMS contém uma rotina $\omega(s, d, k, m)$ que obtém o $k-$menor caminho entre os nós de origem e destino da requisição de tráfego (s e d), para a topologia virtual de modulação m. A rotina $\omega(s, d, k, m)$ retorna um caminho composto pelos nós de origem e destino, s e d, e, possivelmente, nós de articulação adicionais entre eles. Cada travessia em cada nó de articulação representa uma conversão OEO ou, em outras palavras, um salto na

topologia virtual. Consequentemente, a rotina ω retorna um caminho, em que o número de nós deste caminho, menos 1, representa o número de execuções de um algoritmo RSA necessárias para atender a demanda. Portanto, o AMMS permite resolver o problema RMLSA através de $n + 1$ execuções de qualquer algoritmo RSA, onde n é o número de nós de articulação definidos pela rotina ω.

Com o intuito de escolher um nível de modulação apropriado à demanda, um número específico de nós de articulação deve ser calculado, tendo em vista que muitos nós de articulação significam muitos saltos virtuais, o que poderá prejudicar a qualidade da solução. Para este fim, o AMMS define uma constante denominada MHC (*Multi-Hop Constraint*). O MHC é um mecanismo de controle que define o número apropriado de saltos virtuais que o esquema AMMS irá propor. Além disso, o MHC também impacta na escolha do nível de modulação apropriado para a solução RMLSA. Para definir o valor do MHC são considerados a topologia de rede e os níveis de modulação adotados na rede, como mostra a Equação 5.

$$MHC = \left\lceil \frac{dia \times 0,25}{Reach(maxM)} \right\rceil + 1 \tag{5}$$

onde "dia" é o diâmetro da rede e "$Reach(maxM)$" representa o alcance máximo do nível de modulação mais eficiente espectralmente na rede. Por último, a constante $0,25$ foi empiricamente definida através de análises de simulação considerando várias topologias e cenários de rede[1].

Portanto, o MHC define um limite superior no número de nós de articulação. Mais especificamente, $MHC - 1$ representa o número máximo de saltos virtuais que o AMMS deve propor para que se possa aceitar uma requisição de tráfego. Se a rotina $\omega(s, d, k, m)$ retornar um caminho na topologia virtual de modulação com um número de nós maior que MHC, então o esquema irá ignorar este caminho para não oferecer uma solução com muito saltos na topologia virtual. O esquema *AMMS* pode ser visto na Figura 2.

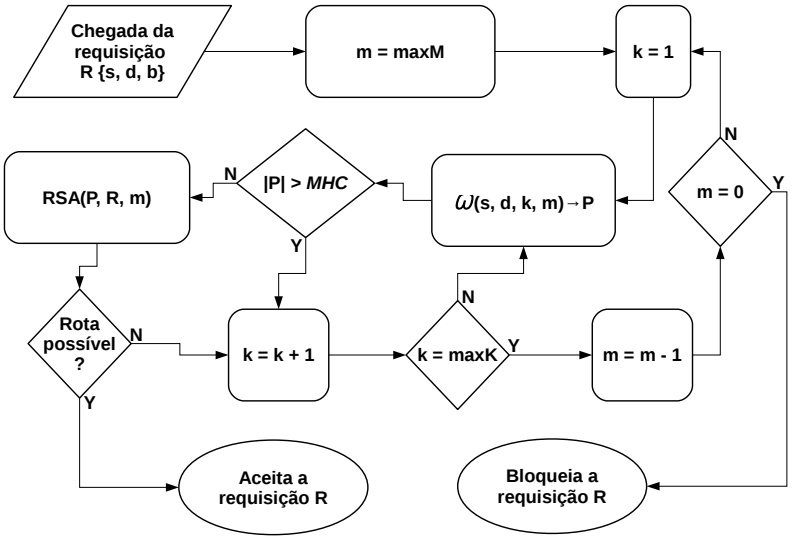


Figura 2. Esquema de modulação adaptativa AMMS.

[1]Os resultados desta avaliação são extensos e portanto foram omitidos neste artigo.

A cada chegada de requisição $R\{s, d, b\}$, o esquema escolhe o nível de modulação mais espectralmente eficiente disponível na rede $R\{s, d, b\}$, em seguida, atribui $k = 1$ e executa a rotina $\omega(s, d, k, m)$, baseado no $k-$menor caminho e no nível de modulação m, definindo o caminho "P". Em seguida, o algoritmo verifica se o número de nós em "P" é maior que "MHC". Se não for, o caminho "P" é enviado para o algoritmo RSA para que ele possa resolver o problema RSA salto a salto em "P" com o nível de modulação "m". Se o algoritmo RSA atribuir espectro para todos os sub-caminhos de "P", então a requisição R é aceita na rede. Caso contrário, ou se $|P| > MHC$ (passo anterior), incrementa-se o valor de k. O próximo passo verifica se $k = maxK$, se não for, a rotina $\omega(s, d, k, m)$ é executada novamente com um novo valor de k, caso contrário decrementa-se m, atribui $k = 1$ e executa-se os passos anteriores novamente. Este laço é executado até $m = 0$, em caso de insucesso, bloqueia-se a requisição.

A complexidade de tempo da fase off-line do esquema AMMS é analisada como segue. Para construção das topologias de modulação tem-se $|V|^2$ execuções do algoritmo de Dijkstra para cada par de nós. Logo a complexidade de tempo é dada como $O(|V|^2 * (|E| + |V|log|V|))$, onde E é o conjunto de enlaces e V é o conjunto de nós da rede. Para formar a estrutura de dados que armazena os $k-$menores caminhos entre cada par de nós em cada topologia virtual de modulação, o algoritmo de Yen [Yen 1971] é utilizado. Dessa forma, a complexidade de tempo final da fase off-line é dada como $O(m * |V|^2 * (k * |V|(E + |V|logV)))$. A complexidade de tempo da fase on-line do esquema AMMS é baseada na quantidade de execuções do algoritmo RSA considerado. A rotina $\omega(s, d, k, m)$ pode ser executada até $maxK * maxM$ vezes e cada execução fornece até V execuções do algoritmo RSA adotado. Logo, a complexidade de tempo do esquema AMMS é dado como $O(V * maxK * maxM)$ multiplicado pela complexidade do algoritmo RSA adotado. É importante observar que $maxK$ e $maxM$ são, na prática, constantes de valor baixo.

5. Avaliação de Desempenho

As simulações foram realizadas com o objetivo de avaliar o desempenho do esquema AMMS proposto em comparação ao esquema *mAdap*. Para ambos os esquemas de modulação foram comparados quatro algoritmos RSA, são eles: KSP, MSP e SPV, utilizando agregação elétrica e o FPA, utilizando agregação elétrica e óptica. Todos os algoritmos possuem o roteamento *single-hop*.

Os algoritmos KSP, MSP e SPV são referenciados em [Wan et al. 2012]. O KSP é o algoritmo de alocação de espectro clássico baseado no roteamento de k-menores caminhos. O MSP é baseado no algoritmo de Dijkstra com incorporação de métodos de alocação de espectro no seu processo. O SPV consiste em um algoritmo de busca e construção de uma árvore para encontrar o menor caminho com espectro disponível para atender à demanda. O FPA foi proposto em [Khodashenas et al. 2013] e utiliza caminhos disjuntos para realizar agregação de trafego e k-menores caminhos para estabelecer um novo caminho óptico. Para todos os algoritmos, foi considerado $k = 3$ e a politica de alocação de espectro First Fit (FF).

5.1. Cenário e Parâmetros Adotados

Foram realizadas simulações usando o simulador de redes ONS [Costa et al. 2016a]. Cada simulação foi realizada cinco vezes utilizando o método de replicações independentes. Para os resultados apresentados foram calculados intervalos de confiança com

95% de confiabilidade. Em cada simulação foram geradas 10^5 requisições de conexão com 6 níveis de granularidade: 25 Gb/s, 50 Gb/s, 100 Gb/s, 200 Gb/s, 300 Gb/s e 400 Gb/s com as proporções 6:5:4:3:2:1, respectivamente. O processo de chegada das chamadas segue a distribuição de *Poisson* com origem e destino distribuídos uniformemente para todos os pares de comunicação da rede.

As topologias consideradas nas simulações foram a USANet com 24 nós e 43 enlaces bidirecionais, e a topologia PanEuro com 27 nós e 81 enlaces bidirecionais. A Figura 3 apresenta as distâncias dos enlaces em quilômetros. A largura de cada *slot* considerada foi $12,5$ GHz e foi assumido que cada enlace possui a capacidade de 320 *slots*. Assume-se uma banda de guarda de 2 *slots*. Cada nó na topologia é equipado com transmissores e receptores suficientes sendo cada transmissor capaz de transmitir até 32 *slots*. As modulações consideradas são BPSK, QPSK, 8QAM e 16QAM com 1, 2, 3 e 4 bits por símbolo e, considerando eventuais regeneradores de sinal, calculam-se as distâncias máximas como 8000, 4000, 2000 e 1000 km, respectivamente.

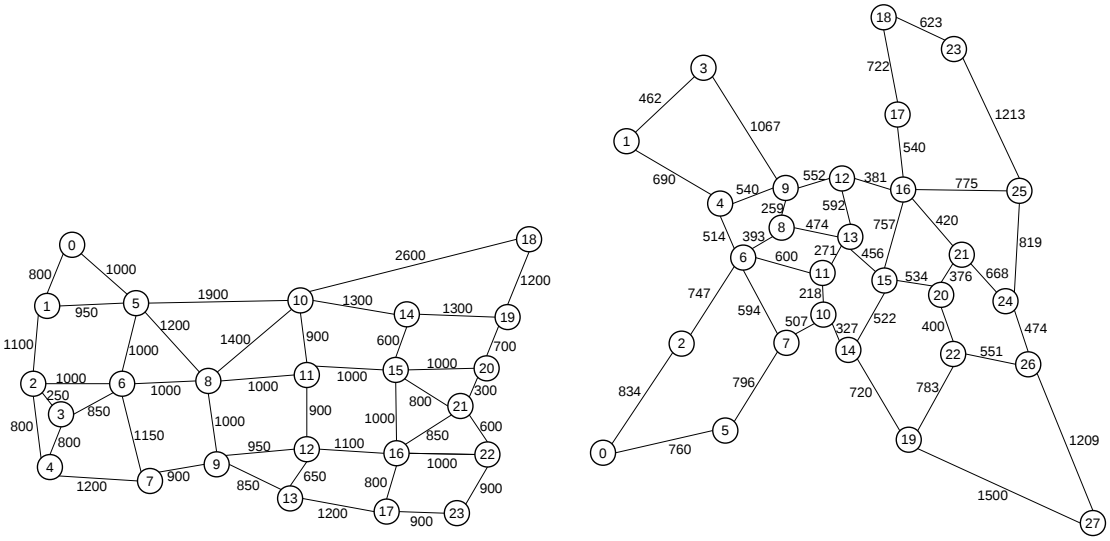


Figura 3. Topologia USANet e PanEuro.

5.2. Resultados Numéricos

As métricas que se referem à utilização de recursos são avaliadas para demonstrar os ganhos relativos aos mecanismos implementados. Além disso, soluções de engenharia de tráfego que levam à redução na utilização dos recursos instalados na rede, auxiliam na diminuição de sua probabilidade de exaustão. As seguintes métricas são avaliadas para todos os algoritmos em ambos os esquemas de modulação adaptativa: Taxa de bloqueio de banda (BBR); média de saltos na topologia virtual por requisição; média da taxa de espectro disponível; taxa média de fragmentação externa; taxa média de consumo de energia; e taxa média do uso de modulação.

Taxa de bloqueio de banda (BBR)

O BBR (Figura 4) mostra a taxa do bloqueio de banda na rede, quanto maior for o BBR, maior será a banda bloqueada na rede e pior o desempenho do algoritmo. Observa-se que em ambas as topologias o desempenho de todos os algoritmos sob o esquema AMMS obteve uma melhora significativa em relação ao esquema de modulação *mAdap*, em média 82% para a USANet e 63% para a PanEuro, para todos os algoritmos RSA compara-

dos. Nota-se que a estratégia tomada pelo esquema *AMMS*, de controle de modulação associado ao controle do número de saltos na topologia virtual, traz aos algoritmos mais possibilidades para atender as demandas, tendo em vista ainda que os requisitos de continuidade de rota são aliviados com o uso dessa estratégia.

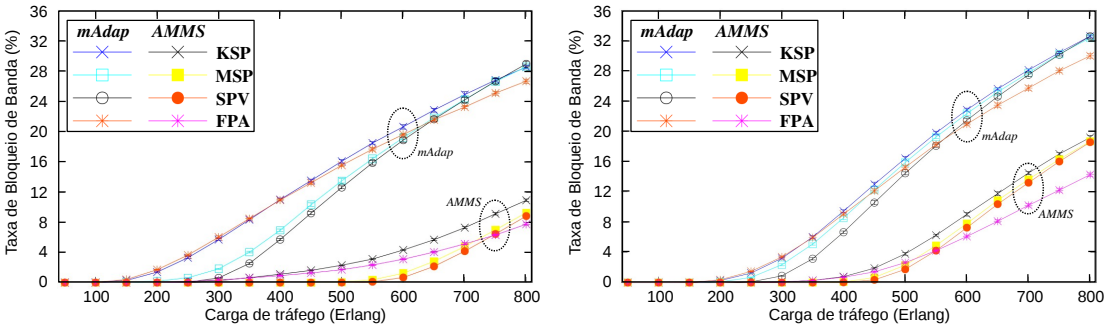


Figura 4. Taxa de bloqueio de banda (BBR). Topologias USANet e PanEuro, respectivamente

Média de saltos na topologia virtual por requisição

O número de saltos na topologia virtual (Figura 5) indica o número de conversões OEO e processamento elétrico que são utilizados na rede. Os algoritmos KSP, MSP, SPV e FPA sob o esquema de modulação adaptativa *mAdap* mantêm em todas as cargas de tráfego uma média igual a 1, uma vez que esses algoritmos são *singlehop*. Observa-se que os algoritmos sob o esquema de modulação adaptativa *AMMS* apresentam médias $1,75$, em ambas as topologias mesmo com o aumento da carga. Estes resultados evidenciam a capacidade do AMMS de assegurar uma solução com poucos saltos virtuais, evitando o atendimento das requisições de tráfego com um número excessivo de conversões OEO. Isso atesta a eficácia do MHC, que limita o uso do roteamento multi-hop contribuindo para diminuição da latência total da rede.

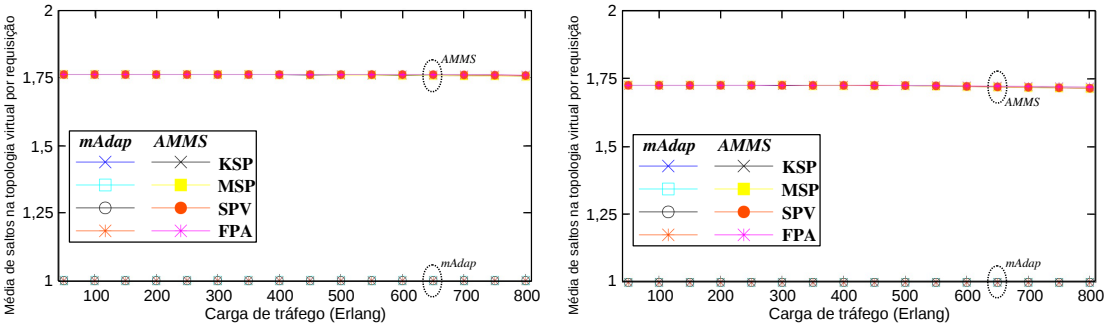


Figura 5. Média de saltos na topologia virtual por requisição. Topologias USANet e PanEuro, respectivamente

Taxa média de espectro disponível

A taxa de espectro disponível (Figura 6) mostra os recursos de espectro utilizados em toda a rede, quanto maior a taxa, mais espectro disponível existe na rede, ou seja, *slots* livres. Todos os algoritmos apresentaram um uso de espectro semelhante em ambos os esquemas de modulação adaptativa, AMMS e *mAdap*, concentrando seu uso em cerca de um terço do total de espectro disponível em toda a rede. Para cargas mais baixas

o AMMS apresentou um leve ganho, cerca de $6,22\%$ para a USANet e $4,14\%$ para a PanEuro. Isso se deve a característica do esquema AMMS, que procura utilizar os níveis de modulação mais espectralmente eficientes da rede e, por sua vez, provocando uma maior economia no uso do espectro. Com o aumento da carga, o uso do espectro aproxima do esquema *mAdap*. Isto acontece pois, devido a limitação de recursos, o AMMS procura criar mais caminhos ópticos na rede aumentando o k. Entretanto observa-se que essa característica não afeta negativamente o desempenho do esquema, tendo em vista que, com mais caminhos ópticos alocados na rede, aumenta-se a oportunidade da utilização de agregação de tráfego na rede.

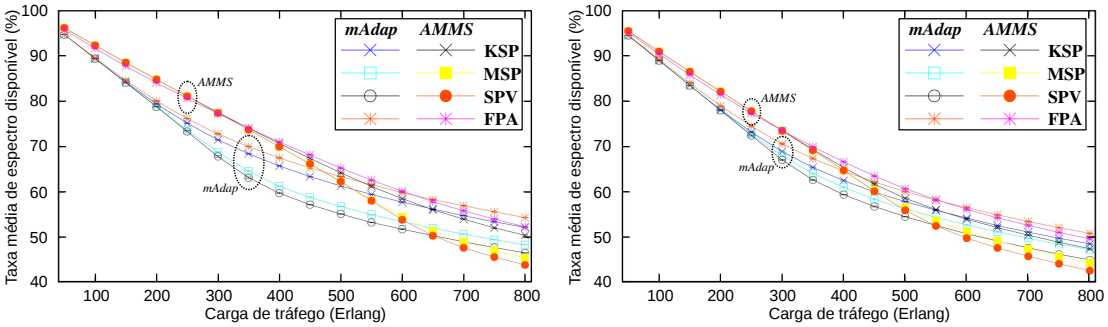


Figura 6. Taxa média de espectro disponível. Topologias USANet e PanEuro, respectivamente

Taxa média de Fragmentação Externa

A taxa média de fragmentação externa (Figura 7) mede o nível de fragmentação externa médio de todos os enlaces da rede durante o tempo de simulação, de acordo com a Equação 1. Níveis elevados representam uma rede com espectro fragmentado e consequentemente provocando uma baixa utilização de recursos de espectro. Observa-se que em ambas as topologias a taxa de fragmentação dos algoritmos sob o esquema AMMS é menor que *mAdap*, apresentando ganhos de cerca de 19% para a USANet e 13% para a PanEuro, para todos os algoritmos RSA. Isso significa que as estratégias tomadas pelo AMMS faz com que os algoritmos RSA fragmentem menos a rede. Isso se deve ao fato do AMMS utilizar níveis de modulação robustos, que carregam mais *bits* por símbolo, e por sua vez formam canais menores com alta capacidade. Canais mais estreitos impactam menos a fragmentação do espectro.

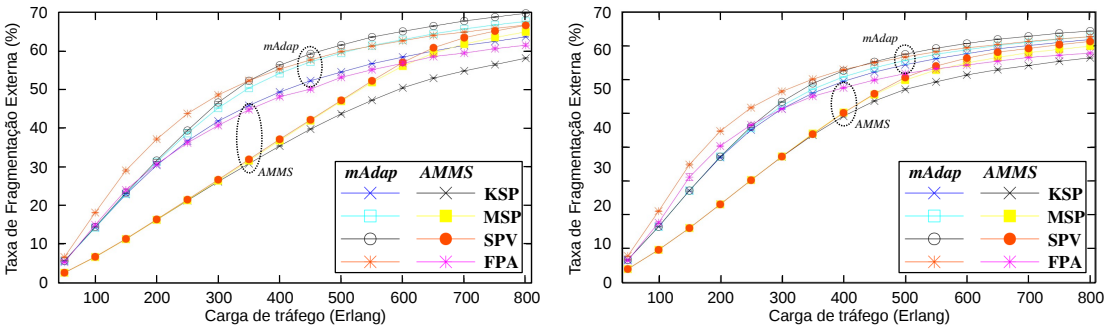


Figura 7. Taxa média de fragmentação externa. Topologias USANet e PanEuro, respectivamente

Taxa média de Consumo de Energia

A taxa média de consumo de energia (Figura 8) mostra a razão entre o gasto total de energia e a quantidade de caminhos ópticos alocados na rede. O modelo energético adotado nos cálculos foi o apresentado na Seção 2.1. É necessário destacar que esta métrica não representa o gasto energético total da rede, mas o gasto médio por caminho óptico alocado. Tendo em vista que o AMMS aloca mais caminhos ópticos que o *mAdap*, pois implementa o roteamento *multi-hop*, observa-se que os caminhos ópticos são mais eficientes e melhores aproveitados energeticamente no esquema AMMS, em média 34% na USANet e 30% na PanEuro.

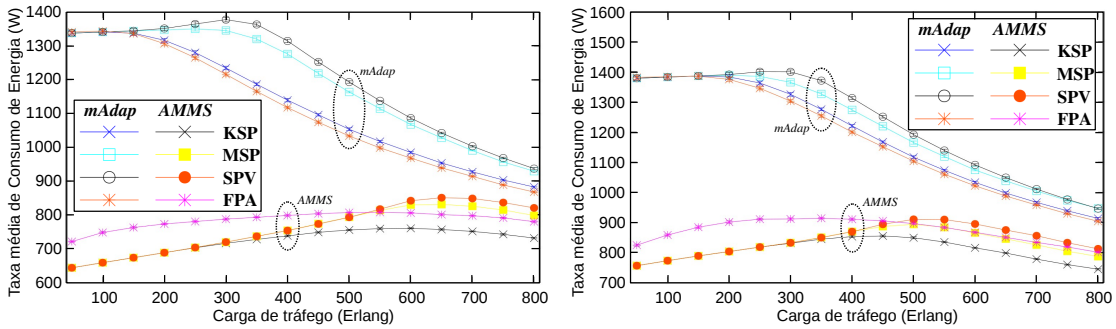


Figura 8. Taxa média de consumo de energia. Topologias USANet e PanEuro, respectivamente

Taxa média do uso de modulação

A taxa média do uso de modulação mostra o percentual de caminhos ópticos alocados em toda a simulação para cada nível de modulação. As Tabelas 2 e 3 mostram a taxa média do uso de modulação considerando a média para todas as cargas de trabalho nas topologias USANet e PanEuro, respectivamente. É necessário destacar que a taxa apresentou uma pequena variação entre as diversas cargas de trabalho, com desvios médios inferiores a $1,2$. Os resultados mostram que, em geral, o esquema de modulação adaptativa AMMS faz com que os algoritmos possam aproveitar ainda mais o uso do nível de modulação mais eficiente espectralmente da rede.

O algoritmo FPA sob o esquema *AMMS* elevou o uso da modulação 8QAM em média cerca de 80%. Para o nível de modulação 16QAM esse aumento foi ainda maior em comparação ao esquema *mAdap*, cerca de 77% para a topologia USANet e 140% para a topologia PanEuro. Observa-se que o AMMS nem sempre usa o melhor nível de modulação da rede. Isso ocorre, pois o esquema enfrenta o compromisso de equilibrar o uso dos formatos de modulação mais espectralmente eficientes e a limitação na quantidade de saltos virtuais necessários para atender a requisição de tráfego.

6. Conclusão

Este trabalho estudou o problema de roteamento e atribuição de espectro com modulação adaptativa (RMLSA). Foi proposta uma técnica para solucionar o problema RMLSA através do uso de um esquema de modulação adaptativa que permite a utilização de qualquer abordagem RSA clássica para resolver o problema RMLSA. O esquema AMMS proposto, procura atribuir níveis de modulação adequados, associado a uma quantidade de saltos necessários para atender uma requisição de tráfego. Dessa forma, o AMMS

Tabela 2. Taxa média do uso de modulação para a topologia USANet (%).

	Modulation	KSP	MSP	SPV	FPA
mAdap	*BPSK*	23,45	24,88	25,62	23,08
	QPSK	42,34	41,39	40,99	41,22
	8QAM	21,72	21,40	21,17	22,98
	16QAM	12,49	12,33	12,22	12,72
AMMS	*BPSK*	0,11	0,14	0,01	0,04
	QPSK	35,90	36,41	36,60	36,26
	8QAM	41,33	41,01	41,00	41,21
	16QAM	22,66	22,44	22,39	22,49

Tabela 3. Taxa média do uso de modulação para a topologia PanEuro (%).

	Modulation	KSP	MSP	SPV	FPA
mAdap	*BPSK*	2,74	3,16	3,45	2,66
	QPSK	42,58	42,34	42,85	41,40
	8QAM	36,05	35,86	35,32	37,25
	16QAM	18,63	18,65	18,38	18,70
AMMS	*BPSK*	0,01	0,02	0,01	0,01
	QPSK	5,15	5,23	5,01	5,08
	8QAM	50,07	50,11	50,28	50,33
	16QAM	44,78	44,64	44,71	44,58

promove o roteamento da requisição através de múltiplos saltos na topologia virtual suavizando as restrições de continuidade de espectro e distância de transmissão, viabilizando o uso de níveis de modulação mais elevados, ao mesmo tempo que permite um melhor aproveitamento dos recursos de espectro na rede.

Para demonstrar a eficiência do esquema de modulação adaptativa proposto, foram comparados quatro algoritmos da literatura sob os esquema de modulação adaptativa *mAdap* de [Wan et al. 2012] e AMMS em duas topologias de rede. Os resultados mostraram que o esquema de modulação adaptativa proposto proporciona um ganho de até 82% na taxa de bloqueio de banda utilizando até $6,22\%$ menos espectro na rede. Observa-se que o AMMS ainda proporciona uma eficiência energética de até 34% em comparação com abordagens da literatura.

Referências

Chatterjee, B., Sarma, N., and Oki, E. (2015). Routing and spectrum allocation in elastic optical networks: A tutorial. *IEEE Communications Surveys Tutorials*, 17(3):1776–1800.

Christodoulopoulos, K., Tomkos, I., and Varvarigos, E. (2011). Elastic bandwidth allocation in flexible ofdm-based optical networks. *Journal of Lightwave Technology*, 29(9):1354–1366.

Costa, L. and Drummond, A. (2016a). Novo Algoritmo RMLSA com Roteamento Multihop em Redes ópticas Elásticas. In *SBRC 2016*, Salvador, Bahia.

Costa, L. R., de Sousa, L. S., de Oliveira, F. R., da Silva, K. A., Júnior, P. J. S., and Drummond, A. C. (2016a). ONS: Optical Network Simulator - WDM/EON. http://comnet.unb.br/br/grupos/get/ons.

Costa, L. R. and Drummond, A. C. (2016b). New distance-adaptive modulation scheme for elastic optical networks. *IEEE Communications Letters*, PP(99):1–1.

Costa, L. R., Ramos, G. N., and Drummond, A. C. (2016b). Leveraging adaptive modulation with multi-hop routing in elastic optical networks. *Computer Networks*, 105:124 – 137.

Horota, A. K., Figueiredo, G. B., and da Fonseca, N. L. S. (2014). Algoritmo de roteamento e atribuição de espectro com minimização de fragmentação em redes Ópticas elásticas. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 32:895–908.

Jinno, M., Kozicki, B., Takara, H., Watanabe, A., Sone, Y., Tanaka, T., and Hirano, A. (2010). Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network [topics in optical communications]. *IEEE Communications Magazine*, 48(8):138–145.

Khodashenas, P. S., Comellas, J., Spadaro, S., and Perelló, J. (2013). Dynamic source aggregation of sub-wavelength connections in elastic optical networks. *Photonic Network Communications*, 26:2–3.

Nag, A., Tornatore, M., and Mukherjee, B. (2010). Optical network design with mixed line rates and multiple modulation formats. *Journal of Lightwave Technology*, 28(4):466–475.

Tomkos, I., Azodolmolky, S., Sole-Pareta, J., Careglio, D., and Palkopoulou, E. (2014). A tutorial on the flexible optical networking paradigm: State of the art, trends, and research challenges. *Proceedings of the IEEE*, 102(9):1317–1337.

Vizcaíno, J. L., Ye, Y., and Monroy, I. T. (2012). Energy efficiency analysis for dynamic routing in optical transport networks. In *2012 IEEE International Conference on Communications (ICC)*, pages 3009–3014.

Wan, X., Hua, N., and Zheng, X. (2012). Dynamic routing and spectrum assignment in spectrum-flexible transparent optical networks. *IEEE/OSA Journal of Optical Communications and Networking*, 4(8):603–613.

Wan, X., Wang, L., Hua, N., Zhang, H., and Zheng, X. (2011). Dynamic routing and spectrum assignment in flexible optical path networks. In *Optical Fiber Communication Conference and Exposition (OFC) and the National Fiber Optic Engineers Conference (NFOEC)*, pages 1–3.

Ye, Z., Patel, A., Ji, P., and Qiao, C. (2014). Distance-adaptive and fragmentation-aware optical traffic grooming in flexible grid optical networks. In *2014 OptoElectronics and Communication Conference and Australian Conference on Optical Fibre Technology*, pages 355–356.

Yen, J. Y. (1971). Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716.

Zhang, G., Leenheer, M. D., Morea, A., and Mukherjee, B. (2013). A survey on ofdm-based elastic core optical networking. *IEEE Communications Surveys Tutorials*, 15(1):65–87.

Zhang, G., Leenheer, M. D., and Mukherjee, B. (2012). Optical traffic grooming in ofdm-based elastic optical networks [invited]. *IEEE/OSA Journal of Optical Communications and Networking*, 4(11):B17–B25.

Zhang, Y., Zheng, X., Li, Q., Hua, N., Li, Y., and Zhang, H. (2011). Traffic grooming in spectrum-elastic optical path networks. In *Optical Fiber Communication Conference and Exposition (OFC) and National Fiber Optic Engineers Conference (NFOEC)*, pages 1–3.

Proteção de Redes Ópticas Elásticas com Multiplexação Espacial Baseada em Modulação, p-Cycle FIPP e Interferência Mínima

Helder M. N. da S. Oliveira[1], **Nelson L. S. da Fonseca**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Campinas 13083-852, SP, Brasil

helder@lrc.ic.unicamp.br, nfonseca@ic.unicamp.br *

Abstract. *Network traffic has grown at an exponential rate and this trend requires large network capacity, lower power consumption and efficient protection techniques. In elastic optical networks, Routing, Modulation, and Spectrum Allocation algorithms have received attention as schemes to protect traffic flows against network failures. However, it has not been studied for elastic optical networks using spatial division multiplexing (SDM). This paper introduces a new algorithm to provide path protection using p-cycle, minimal interference, and modulation for elastic optical networks with spatial division multiplexing. The proposed algorithm is compared to other algorithms in the literature. Results indicate that the proposed algorithm provides 100% protection against single failures under low loads.*

Resumo. *O tráfego de redes tem crescido a uma taxa exponencial e esta tendência exigirá das redes do futuro grande capacidade, menor consumo de energia e técnicas de proteções eficientes. Os algoritmos de roteamento e atribuição de espectro, com modulação adaptativa para redes ópticas elásticas, têm sido investigado como esquemas para proteger o tráfego contra falhas de rede. Entretanto, não foi ainda estudado para redes ópticas elásticas usando multiplexação por divisão espacial (SDM). Este artigo introduz um novo algoritmo para fornecer proteção de caminho, utilizando p-cycle, interferência mínima e modulação para redes ópticas elásticas com multiplexação por divisão espacial. O desempenho do algoritmo proposto é comparado com os algoritmos existentes na literatura. Os resultados indicam que o algoritmo proposto fornece 100% de proteção para falhas individuais sob baixa sobrecarga.*

Introdução

Estima-se que nos próximos dez anos o acesso de usuários finais irá crescer de 100 Mb/s para 1 GB/s, motivado por inovações tecnológicas das fibras ópticas de alta capacidade para o segmento de acesso. Em consequência a esses avanços, o núcleo da rede deverá comutar a 1 Tb/s [Tomkos et al. 2013]. O problema é que uma fibra de único núcleo, fibra amplamente utilizada nas redes ópticas existentes, possui capacidade física limitada e há uma tendência dessa capacidade não ser suficiente em um futuro próximo [Essiambre et al. 2010].

A rede óptica elástica (EON) com multiplexação por divisão de espaço (SDM) sobre fibra multi-núcleo (MCF) é uma solução promissora para superar as demandas heterogêneas de banda passante das aplicações. A capacidade de se alocar flexivelmente

o espectro, permitirá que estas redes lidem com demandas de tráfego com requisitos de banda variando de sub-comprimento de onda a super-comprimento de onda. Além disso, a adoção de SDM irá proporcionar as redes maior capacidade e eficiência espectral.

O uso de múltiplos núcleos nas fibras é uma forma eficiente de concepção e fabricação para utilização de SDM. A adoção destas fibras ópticas suportando SDM aumenta a capacidade das fibras por um fator igual ao número de cores espaciais na fibra.

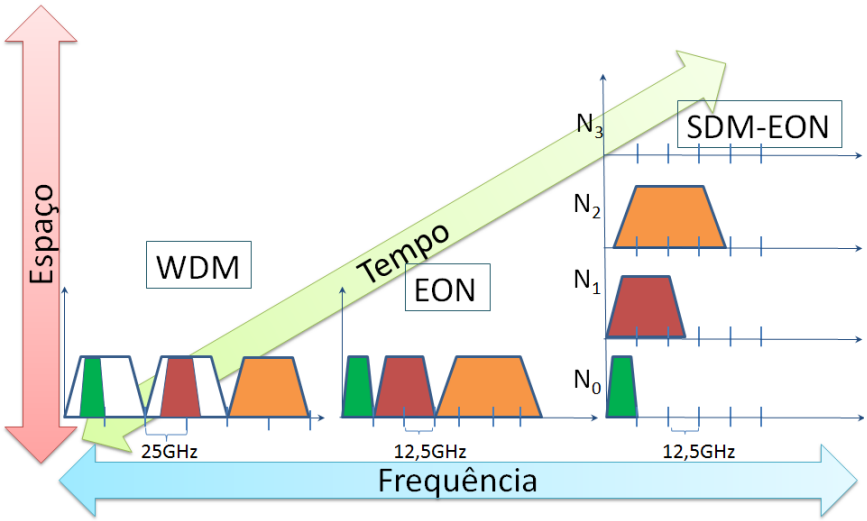


Figura 1. Três possibilidades diferentes para provisionamento de recursos.

A Figura 1 ilustra uma rede óptica com multiplexação por divisão de comprimento de onda (WDM), uma rede óptica elástica convencional (EON) e uma rede óptica elástica utilizando multiplexação espacial (SDM-EON). Na figura, para rede SDM-EON, assume-se que existe um sistema SDM com quatro núcleos. A rede WDM tem menor capacidade de tráfego que a rede EON. Em redes WDM, um comprimento de onda pode acomodar demandas de diferentes tamanhos, subutilizando a largura de banda disponível. Em uma rede óptica elástica, considerando-se uma modulação de 4 bits por símbolo e a tecnologia SDM, para uma conexão de 1 Tb/s entre dois nós, a flexibilidade proporcionada pelo uso de três dimensões (temporal, espectral e espacial) permite que esta conexão seja estabelecida criando um supercanal que se estende pelos quatro núcleos com cinco *slots* espectrais de 12,5 GHz em cada núcleo.

Além dos efeitos físicos presentes em EON (acumulação de ruído óptico, mistura de quatro ondas), a dimensão espacial pode introduzir uma quantidade significativa de interferência de banda, devido ao modo de acoplamento e *crosstalk* (XT) nos elementos MUX/DEMUX espaciais distribuídas ao longo dos enlaces de fibra. Uma questão a ser considerada na criação de rotas em redes ópticas com multiplexação espacial é o *crosstalk* entre núcleos. *Crosstalk* entre núcleos acontece quando sinais se propagam no mesmo espectro em núcleos adjacentes em fibras multi-núcleo (MCF).

Apesar da capacidade de sobrevivência ser de suma importância em redes que transportam uma grande quantidade de tráfego, a maioria dos estudos sobre EONs não considera proteção dessas redes. As redes ópticas elásticas com multiplexação por divisão

de espaço (SDM) sobre fibra multi-núcleo possuirá maior capacidade de banda que as redes com apenas um canal, havendo uma maior necessidade de proteção.

A vulnerabilidade de redes ópticas tem motivado o desenvolvimento de diferentes esquemas de proteção e restauração. O p-cycle é uma dessas técnicas de proteção, que tem sido intensivamente investigado nos últimos anos devido às suas vantagens. O *p-cycle* é um esquema de proteção em que a capacidade reservada é pre-conectada e forma estruturas em anel para proteção de redes em malha [Asthana et al. 2010]. *p-Cycles* fornecem proteção semelhante a proteção fornecida por *Bidirectional Line Switched Ring*, que é considerado uma generalização do esquema de proteção 1:1, ou seja, um anel protegendo um ciclo. A diferença fundamental entre *p-cycle* e proteção em anel é a proteção de enlaces transzonais, que são enlaces que não estão no anel (ciclo) e cujos os dois nós finais estão no ciclo. Um caso especial de *p-cycle* é o *p-cycle* FIPP (caminho de proteção independente de falhas). *p-Cycle* FIPP é uma extensão do conceito de *p-cycle* que permite que falhas não sejam necessariamente limitadas a um enlace ou segmento de caminho imediatamente adjacente aos nós finais. FIPPs proporcionam a vantagem da detecção de falha ser independente de sua localização, diz-se, então, que é "independente de falha". Tal propriedade é vantajosa quando a localização da falha é lenta ou difícil, como em redes transparentes e translúcidas. Adicionalmente, um *p-Cycle* FIPP é capaz de fornecer a velocidade de restauração rápida mantendo a eficiência de espectro [Kodian and Grover 2005].

Os algoritmos tradicionais de proteção levam a uma rápida saturação dos enlaces de rede, o que motiva a concepção de novos algoritmos, especialmente aqueles que empregam interferência mínima para promover utilização equilibrada dos recursos. Algoritmos de interferência mínima geram conexões que interferem menos com pedidos de entrada para estabelecimento de conexão [Figueiredo et al. 2004, Figueiredo et al. 2006]. Ao proteger um caminho em falha, *p-cycles* podem sobrecarregar enlaces, uma vez que, o *p-cycle* pode usar os mesmos enlaces que os caminhos primários. A idéia é gerar *p-cycle* transzonal ao caminho primário, impedindo *p-cycles* e caminhos de usarem os mesmos enlaces, portanto, minimizando a rejeição de requisições futuras.

O problema de roteamento e alocação de espectro (RSA) é um problema fundamental no projeto de redes EONs. O problema de Roteamento, Nível de Modulação e Atribuição de Espectro (RMLSA) inclui a escolha entre os diferentes formatos de modulação. Os de algoritmos RMLSA atribuem slots contínuos e contíguos em todos os enlaces do caminho selecionado, além de escolher um nível modulação adequado. A utilização de diferentes formatos de modulação pode obter velocidades de transmissão elevadas. No entanto, a transmissão de um número elevado de bits por símbolo depende da qualidade de transmissão (Qot), que por sua vez sofre grande influência do comprimento do caminho, uma vez que a distância afeta a capacidade do receptor de decodificar o sinal recebido [Moura et al. 2015]. A combinação de SDM e formatos de modulação permite uma ganho na utilização do espectro, o que diminui o bloqueio das requisições. Esse problema é chamado Roteamento, Nível de Modulação, Núcleo e Alocação do Espectro (RMLCSA).

Apesar de outros trabalhos tratarem de proteção em redes óptica elásticas, nenhum dos trabalhos na literatura empregam modulação adaptativa, p-cycle e interferência mínima para proteção de caminhos em redes óptica com multiplexação espacial. Este artigo introduz o algoritmo Modulation, Minimum Interference and Failure-

independent path protecting for MultiCore network(MMIFMC) para prover proteção para redes ópticas elásticas com multiplexação espacial. O algoritmo decide sobre caminhos de proteção, utilizando *p-Cycle* FIPP, interferência mínima e diferente formatos de modulação. O algoritmo MMIFMC prioriza o uso de *p-cycles* transzonais, gera interferência mínima e reduz o número de bloqueio de conexões. Resultados mostram que o algoritmo provê proteção de caminho sem aumentar significativamente bloqueio de requisições para estabelecimento de conexões.

Este artigo está organizado da seguinte forma. A seção 2 apresenta os formatos de modulação utilizados. A seção 3 revisa trabalhos relacionados. A seção 4 introduz o algoritmo MMIFMC. A seção 5 avalia o desempenho do algoritmo proposto e a seção 6 conclui o artigo.

Formatos de Modulação

Em redes ópticas elásticas, seleciona-se o formato de modulação a ser utilizado nos caminhos considerando-se a distância entre os nós de origem e destino. A transmissão adaptativa à distância (DAT), escolhe o formato de modulação considerando apenas a distância de transmissão [Costa et al. 2016, Costa and Drummond 2016]. Neste método, o formato de modulação mais eficiente, do ponto de vista espectral, é selecionado de forma que o comprimento do caminho não exceda o alcance da transmissão.

Neste trabalho, utilizou-se formatos de modulação conforme tabela 1, que mostra a capacidade dos slots de acordo com o formato de modulação [Vizcaíno et al. 2012]. O formato de modulação depende da distância entre os nós de origem e destino.

Tabela 1. Características de Modulação

Formato de Modulação	Bits por Símbolo	Capacidade do Slot (Gb/s)	Distancia Máxima (km)
64QAM	6	75	125
32QAM	5	62.5	250
16QAM	4	50	500
8QAM	3	37.5	1000
QPSK	2	25	2000
BPSK	1	12.5	4000

Trabalhos Relacionados

Algorítimos de roteamento e alocação de núcleo e espectro (RSCA) para redes ópticas elásticas com multiplexação espacial têm sido propostos recentemente na literatura [Muhammad et al. 2014], [Tode and Hirota 2014], [Fujii et al. 2014], [Zhang et al. 2012] e [Klonidis et al. 2015]. No entanto, somente em [Oliveira and Da Fonseca 2016a] e [Oliveira and Da Fonseca 2016b] estudos relacionados à proteção em redes ópticas elásticas com multiplexação espacial são considerados. Além disso, nenhum dos trabalhos mencionados inclui a escolha da modulação na definição de rotas e alocação do espectro.

Em [Oliveira and Fonseca 2016], apresenta-se um algoritmo para fornecer proteção de caminho através do emprego de p-cycle de caminho, agregação de tráfego

e sobreposição de espectro em redes ópticas elásticas. No entanto, multiplexação espacial, mínima interferência e modulação não são empregados.

Em [Moura et al. 2015], propõe-se um algoritmo de roteamento, nível de modulação e atribuição de espectro para redes ópticas elásticas convencionais considerando o consumo de energia do caminho primário.

Os autores em [Jinno et al. 2010] propuseram uma solução de roteamento, nível de modulação e atribuição de espectro usando K caminhos mais curtos para calcular rotas e uma política para alocar espectro usando o menor slot inicial disponível no espectro. A modulação é escolhida com base no comprimento dos caminhos, de forma que utilize uma faixa menor do espectro e possa descodificada com êxito no destino.

Os autores em [Chen et al. 2015] avaliaram a eficiência espectral do p-cycle FIPP em redes ópticas elásticas convencionais juntamente com modulação adaptativa. Foram utilizados cenários estáticos e dinâmicos para avaliação do esquema.

Em [Tode and Hirota 2014], o problema de RSCA é dividido em problema de roteamento e SCA e introduz um método de pré-computação de k-caminhos como uma solução de roteamento.

Os autores em [Muhammad et al. 2014] desenvolveram uma formulação de programação linear inteira (ILP) para o problema de planejamento de rede RSCA, priorizando a eficiência espectral nos núcleos da rede óptica elástica 3D. Eles utilizaram 7 núcleos por fibra em suas simulações.

Dentre todos os trabalhados encontrados, nenhum trata de proteção de redes ópticas elásticas com multiplexação espacial e utilizando diferente formatos de modulação.

O Algoritmo MMIFMC

O algoritmo introduzido nesta subseção, chamado *Modulation, Minimum Interference and Failure-independent path protecting for MultiCore networks* (MMIFMC) decide sob provisão de proteção dos caminhos ópticos através do uso de *p-cycles* FIPP. *p-Cycles* FIPP baseiam-se em diferentes caminhos primário e de *backup*, e apresentam a vantagem de que a detecção de falha é independente da localização da falha. Tal propriedade é vantajosa quando a localização da falha é lenta ou difícil, como em redes transparentes ou translúcidas. O algoritmo MMIFMC decide sobre a criação de caminhos ópticos em uma rede óptica elástica SDM protegida por *p-cycle* FIPP e é apresentado no Algoritmo 1. Neste estudo, um caminho óptico é estabelecido se e somente se ele pode ser protegido por um p-cycle.

O algoritmo MMIFMC garante um caminho de proteção para cada caminho óptico estabelecido e a proteção é garantida para falhas únicas. A reserva de recursos para criar o *p-cycle* FIPP protegendo uma requisição pode sobrecarregar os enlaces na rede, uma vez que o *p-cycle* que protege os caminhos no ciclo usam os mesmos enlaces do caminho primário. Por outro lado, o *p-cycle* que protege caminhos transzonais tendem a reservar mais recursos, pois tem um maior número de saltos. Os *p-cycles* que protegem caminhos no ciclo utilizam menos recursos da rede do que p-cycle que protegem caminhos transzonais, por outro lado, podem sobrecarregar os enlaces ao longo de um caminho. Por conseguinte, é necessário adotar critérios para evitar a formação de gargalos, equilibrando

a carga entre potenciais caminhos; é necessário adotar uma abordagem de interferência mínima para evitar o bloqueio de requisições.

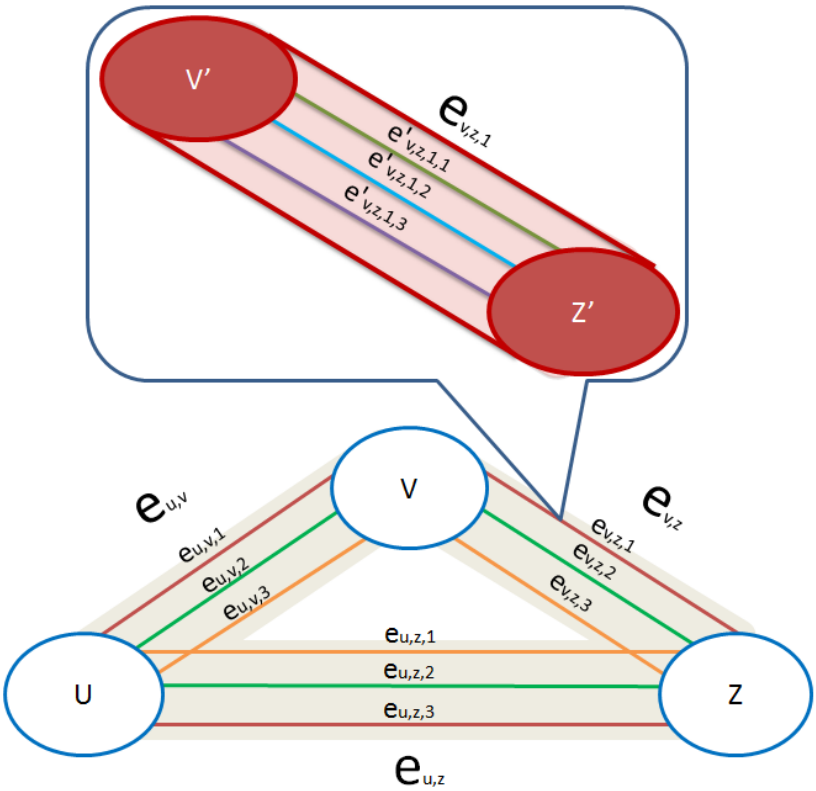


Figura 2. Multigrafo

Os modelos de algoritmo proposto consideram a disponibilidade de espectro na rede, como um multigrafo marcado (Figura 2).

A seguinte notação matemática será usada neste algoritmo:

s: nó fonte;

d: nó destino;

b: demanda de largura de banda;

$r(s, d, b)$: requisição do nó s para o nó d com demanda b em *slots*;

N: número do conjunto de slots entre dois nós;

M: número de possíveis níveis de modulação variando de acordo com a tabela 1;

b_m: relação entre a requisição b e a taxa de nível de modulação m, multiplicado pela capacidade do slot, que é dado pelo número de slots necessário para alocar b usando a modulação m [Moura et al. 2015];

C: número do conjunto de núcleos entre dois nós;

$G = (V, E, W)$: multigrafo marcado composto por um conjunto de nós V, um conjunto de arestas E e um conjunto de pesos de arestas W, $|E| = C \cdot N \cdot |V|$. As arestas conectando dois vértices de G representam N *slots* nos enlaces e C núcleos, conectando

dois nós na rede;

$E = \{e_{u,v,n}\}$: conjunto de n arestas;

$e_{u,v,n}$: A *n-ésima* aresta conectando u e v;

$e'_{u,v,n,j}$: onde *j* será um canal escolhido para obter o menor *crosstalk*, baseado nos slots vizinhos.

$w(e_{u,v,n})$: peso da aresta $e_{u,v,n}$;

$w(e_{u,v,n}) = 1$ se o *n-ésimo slot* no enlace conectando os nós u e v estão livres e $w(e_{u,v,n}) = \infty$ se o *slot* já estiver alocado;

$W = \{w(e_{u,v,n})\}$:conjunto de pesos das arestas;

$\widetilde{G}_{n,b} = (\widetilde{V}, \widetilde{E}, \widetilde{W})$: o n-ésimo grafo marcado tal que $\widetilde{E}$ é o conjunto de arestas conectando $\{\widetilde{u}, \widetilde{v}\} \in \widetilde{V}$ e $\widetilde{W}$ é o conjunto de custos associados a $\widetilde{E}$. A aresta $\widetilde{E}$ corresponde ao mapeamento de b aresta em G iniciando na n-ésima aresta;

$\widetilde{V} = V$: conjunto de nós;

$\widetilde{e}_{u,v} \in \widetilde{E}$: aresta conectando $\widetilde{u}$ e $\widetilde{v}$;

$\widetilde{e}_{\widetilde{u},\widetilde{v}} = \{e_{u,v,n}\} \in E$ é uma sequência tal que $e_{u,v,n}$ é a menor aresta ordenada, $e_{u,v,n+b}$ é a maior aresta ordenada $|\widetilde{e}_{u,v}| = b$;

$\widetilde{w}_n(\widetilde{e}_{\widetilde{u},\widetilde{v}})$: peso da aresta $\widetilde{e}_{\widetilde{u},\widetilde{v}}$;

$\widetilde{W}_n = \{\widetilde{w}_n(\widetilde{e}_{\widetilde{u},\widetilde{v}})\}$: conjunto de pesos das arestas;

P_n: sequência de $\widetilde{G}_n$ tal que o nó fonte s é o menor nó ordenado e d é o maior nó ordenado;

$W(\widetilde{P}_n)$: $\sum_{\widetilde{e}_{\widetilde{u},\widetilde{v}} \in \{\widetilde{P}_n\}} \widetilde{e}_{\widetilde{u},\widetilde{v}}$: o peso do caminho $\widetilde{P}_n$ é a soma dos pesos de todos as arestas na sequência;

$W_{P_{s,d}}$ = peso do menor caminho entre s e d;

$\widetilde{t}_{u,v,b}$: *p-cycle* contendo os vértices u e v e arestas correspondendo ao mapeamento de b arestas do multigrafo G;

$\widetilde{T}_{u,v,b} = \widetilde{t}_{u,v,b}$: conjunto de todos os *p-cycles* contendo os vértices u e v and arestas correspondentes ao mapeamento de b arestas do multigrafo G;

$\widetilde{T}$: conjunto de todos os *p-cycles* ativos;

T_n: sequência de $\widetilde{G}_n$ tal que o nó fonte s é o menor nó ordenado e d é o maior nó ordenado;

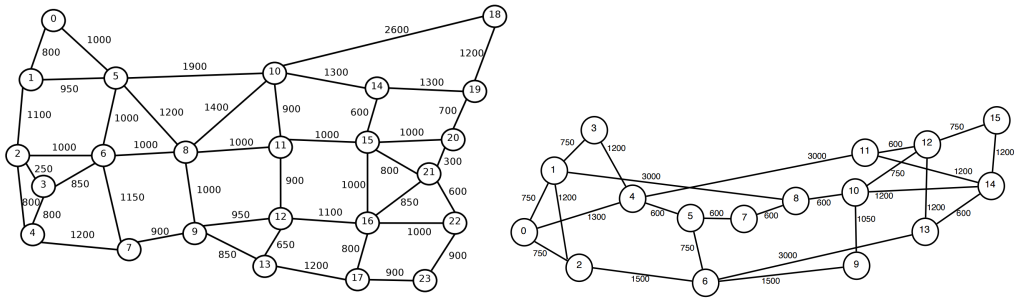
$W(\widetilde{T}_n)$: $\sum_{\widetilde{e}_{\widetilde{u},\widetilde{v}} \in \{\widetilde{T}_n\}} \widetilde{e}_{\widetilde{u},\widetilde{v}}$: o peso do *p-cycle* $\widetilde{T}_n$ (a soma dos pesos de todas as arestas da sequência);

$W_{T_{s,d}}$ = peso do *p-cycle* que protegerá o caminho entre s e d;

Neste algoritmo, a linha 1 estabelece o conjunto de arestas que serão mapeados para $\widetilde{G}_{n,b_m}$ arestas. A linha 2 estabelece todo o conjunto de modulações que será testado. A linha 3 resolve um algoritmo de caminho mais curto para o grafo $\widetilde{G}_{n,b_m}$ e fornece o

Algorithm 1 MMIFMC

1: $\forall m \in M$
2: $\quad \forall n = 1...C(N-b_m)$
3: $\quad\quad (W(P_n), P_n) = MenorCaminho(\widetilde{G}_{n,b_m}, r(s,d,b_m))$
4: $\quad W_{P_{s,d}} = W(P_n) | \, \forall i \; W(P_n) \leq W(P_i)$
5: **if** $W_{P_{s,d}} = \infty$ **then**
6: $\quad block \; r(s,d,b)$
7: **else**
8: $\quad$ **if** $T_n \neq \emptyset \quad \forall \, T_n \in \widetilde{T}$ **then**
9: $\quad\quad$ establish $r(s,d,b)$ as P_n and T_n
10: $\quad\quad W(e'_{u,v,i}) = \infty \quad \forall \{u,v\} \in \widetilde{P}_i \quad n = n...i+b_m - 1$
11: $\quad$ **else**
12: $\quad\quad \forall m \in M$
13: $\quad\quad\quad \forall n = 1...C(N-b_m)$
14: $\quad\quad\quad\quad (W(T_n), T_n) = CicloTranszonal \; (\widetilde{G}_{n,b_m}, r(s,d,b_m))$
15: $\quad\quad W_{T_{s,d}} = W(T_n) | \, \forall i \; W(T_n) \leq W(T_i)$
16: $\quad\quad$ **if** $W_{T_{s,d}} = \infty$ **then**
17: $\quad\quad\quad \forall m \in M$
18: $\quad\quad\quad\quad \forall n = 1...C(N-b_m)$
19: $\quad\quad\quad\quad (W(T_n), T_n) = MenorCiclo \; (\widetilde{G}_{n,b_m}, r(s,d,b_m))$
20: $\quad\quad\quad W_{T_{s,d}} = W(T_n) | \, \forall i \; W(T_n) \leq W(T_i)$
21: $\quad\quad\quad$ **if** $W_{T_{s,d}} = \infty$ **then**
22: $\quad\quad\quad\quad$ block $r(s,d,b)$
23: $\quad\quad\quad$ **end if**
24: $\quad\quad$ **end if**
25: $\quad\quad$ **if** $W_{T_{s,d}} \neq \infty$ **then**
26: $\quad\quad\quad$ establish $r(s,d,b)$ as $\widetilde{P}_n$ and $\widetilde{T}_n$
27: $\quad\quad\quad W(e'_{u,v,i}) = \infty \quad \forall \{u,v\} \in \widetilde{P}_i \quad n = n...i+b_m - 1$
28: $\quad\quad\quad W(e'_{u,v,i}) = \infty \quad \forall \{u,v\} \in \widetilde{T}_i \quad n = n...i+b_m - 1$
29: $\quad\quad$ **end if**
30: $\quad$ **end if**
31: **end if**



(a) USA (b) NSF

Figura 3. Topologias

caminho e o custo dele. Se o peso do caminho mais curto é ∞, não foi possível encontrar um caminho sob a restrição de contiguidade para a demanda b com a alocação começando com o n-ésimo *slot*. A linha 4 seleciona o caminho entre os $C \cdot (N - b + 1)$ caminhos mais curtos que tem o menor peso em conjunto com os respectivos núcleos selecionados. No caso em que o peso do caminho mais curto é ∞ (linha 5), não existe um caminho na rede que atenda a requisição de b *slots*. Portanto, a requisição é bloqueada (Linha 6). Caso contrário, um *p-cycle* para proteger o caminho óptico é pesquisado (linha 8). Caso, exista um *p-cycle*, o caminho óptico é estabelecido (linha 9) e as arestas correspondentes no multigrafo G têm o seu peso alterado para ∞ (linha 10), o que significa que os slots foram alocados para o caminho óptico recém-criada. Caso contrário, um *p-cycle* com mínima interferência para proteger o caminho óptico deve ser criado. A linha 12 busca uma modulação pra o p-cycle que será criado para proteger o caminho. Para evitar gargalos na rede e adotando o conceito de mínima interferência, o algoritmo tenta criar um p-cycle transzonal ao caminho (Linha 13). Caso nenhum *p-cycle* transzonal possa ser criado para proteger o caminho óptico, então o algoritmo tenta criar o menor caminho (Linha 17). Caso nenhum *p-cycle* possa ser criado a requisição é bloqueada (Linha 20). Se o p-cycle for criado (Linhas 23), o p-cycle e o caminho são estabelecidos para satisfazer o pedido e as arestas correspondentes no multigrafo G tenham o seu peso alterado para ∞ (Linha 25 e 26), o que significa que os slots foram alocados para o caminho óptico recém-criado.

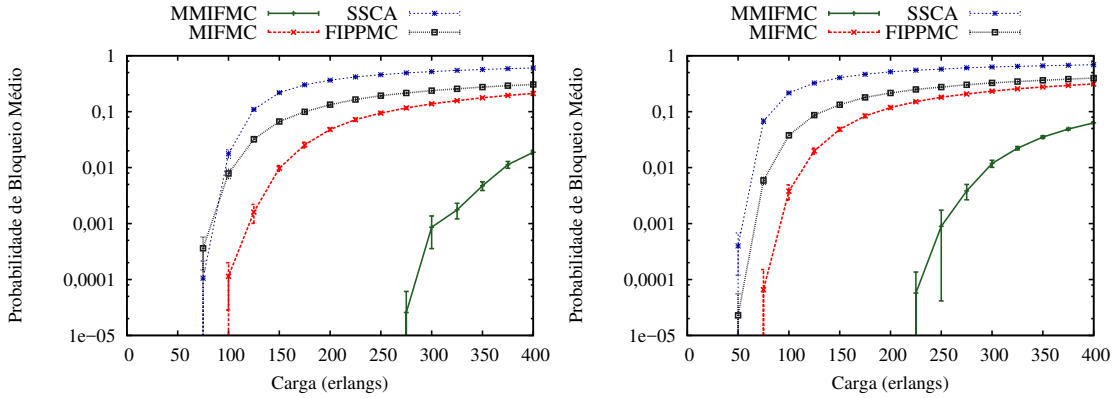
Avaliação de Desempenho

Para avaliar o desempenho do Algoritmo 1, experimentos de simulação foram realizados utilizando o simulador FlexGridSim [Moura and Drummond]. Em cada simulação, 100.000 requisições foram geradas e utilizou-se o método de replicação independente e adotou-se nível de confiança de 95% para os intervalos de confiança. As topologias NSF (Figura 3(b)) com 16 nós e 25 enlaces bidirecionais e USA (Figura 3(a)) com 24 nós e 43 enlaces bidirecionais foram empregadas, as figuras mostram as respectivas distancias entre os nós. O espectro foi dividido em 240 slots de espectro e as fibras simulada possuem 7 núcleos. A carga foi variada entre 25 e 400 erlangs em intervalos de 25 erlangs.

As métricas utilizadas na comparação são relação de bloqueio de banda, relação de crosstalk por slot, número médio de saltos do caminho primário e número médio de saltos do caminho de backup.

Nas figuras, as curvas rotuladas como FIPPMC mostram os resultados para as redes que utilizam o algoritmo FIPPMC [Oliveira and Da Fonseca 2016a], as curvas ro-

tuladas como MIFMC mostram os resultados para as redes que utilizam o algoritmo Minimum Interference and Failure-independent path protecting for MultiCore networks (MIFMC) [Oliveira and Da Fonseca 2016b], as curvas rotuladas como SSCA encontra o caminho primário através do algoritmo proposto em [Tode and Hirota 2014], utilizando $k = 3$ e o caminho de proteção adicionando a capacidade de compartilhamento ao mesmo algoritmo, enquanto as curvas rotuladas como MMIFMC mostram os resultados para as redes que utilizam o algoritmo MMIFMC.



(a) USA (b) NSF

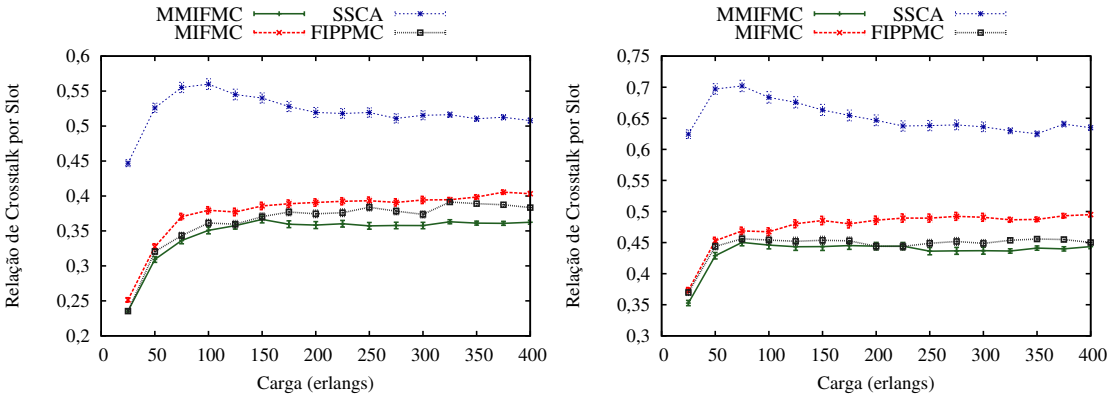
Figura 4. Bloqueio de banda em função da carga da rede

As Figuras 4(a) e 4(b) mostram a probabilidade de bloqueio (*Bandwidth Blocking Ratio*) para a topologia USA e NSF, respectivamente.

Para a topologia USA (Figura 4(a)), devido à alta conectividade da topologia USA não há bloqueio até 75 erlangs. Enquanto os algoritmos FIPPMC e SSCA iniciam o bloqueio sob carga de 75 erlangs. O algoritmo MIFMC inicia o bloqueio sob cargas de 100 erlangs e o algoritmo MMIFMC inicia o bloqueio somente sob cargas 275 erlangs, isto ocorre devido à propriedade de modulação adaptativa que o algoritmo possui, garantindo melhor utilização do espectro. Sob cargas de 125 erlangs, o algoritmo MIFMC possui probabilidade de bloqueio uma ordem de magnitude menor que o FIPPMC e quase duas ordens de magnitude menor que o SSCA, enquanto que o algoritmo MMIFMC que utiliza modulação adaptativa não produz bloqueio. Sob cargas de 250 erlangs, o algoritmo MMIFMC possui probabilidade de bloqueio muito baixa, enquanto que os algoritmos FIPPMC, SSCA e MIFMC esgota os recursos da rede.

Para a topologia NSF (Figura 4(b)), os algoritmos FIPPMC e SSCA inciam o bloqueio sob cargas de 50 erlangs, isso acontece devido a baixa conectividade da topologia. Enquanto os algoritmos FIPPMC e SSCA iniciam o bloqueio sob cargas de 50 erlangs, o algoritmo MIFMC inicia o bloqueio sob cargas de 75 erlangs e o algoritmo MMIFMC inicia o bloqueio somente sob cargas 225 erlangs, isto ocorre devido à propriedade de modulação adaptativa que o algoritmo possui, garantindo melhor utilização do espectro. Sob cargas de 75 erlangs, o algoritmo MIFMC possui probabilidade de bloqueio quase duas ordens de magnitude menor que o FIPPMC e quase quatro ordens de magnitude menor que o SSCA, enquanto que o algoritmo MMIFMC que utiliza modulação adaptativa não produz bloqueio. Sob cargas de 225 erlangs, o algoritmo MMIFMC produz probabilidade de bloqueio muito baixa, enquanto que os algoritmos FIPPMC, SSCA e MIFMC

esgota os recursos da rede.



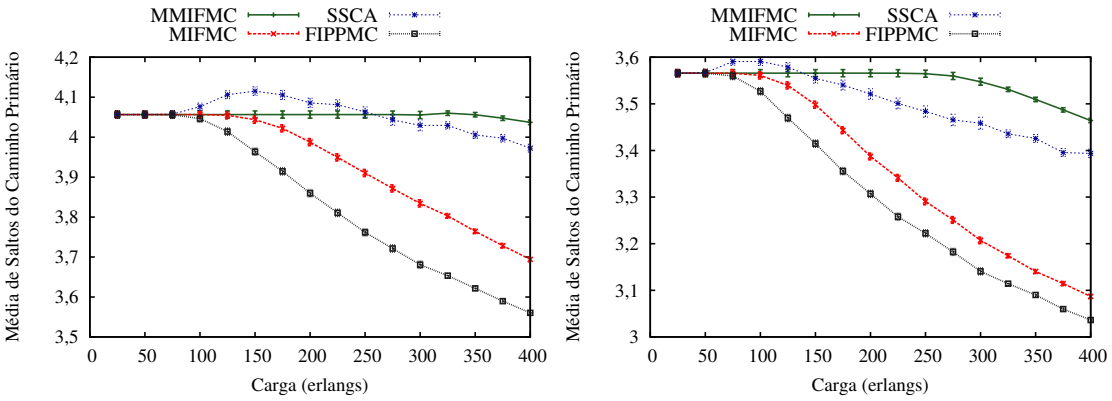
(a) USA (b) NSF

Figura 5. Relação de crosstalk por slot

As Figuras 5(a) e 5(b) representam a relação de *crosstalk* por slot em função da carga da rede para as topologia USA e NSF, respectivamente. O valor de *crosstalk* para cada slot de espectro é definido como a razão entre o índice de *crosstalk* atual e o valor máximo do índice *crosstalk*. A relação de *crosstalk* por slot é definida pelo valor médio entre todos os slots de espectro da rede.

Para a topologia USA (Figura 5(a)), a relação de *crosstalk* gerado pelo algoritmo SSCA inicia em 0,44 e aumenta até 0,55. Por sua vez, os algoritmos MMIFMC, FIPPMC e MIFMC possuem comportamento similar, variando entre 0,23 e aumentando até 0,41.

Para a topologia NSF (Figura 5(b)), a relação de *crosstalk* gerado pelo algoritmo SSCA inicia em 0,62 e aumenta até 0,71. Como na topologia USA, os algoritmos MMIFMC, FIPPMC e MIFMC possuem comportamento similar, variando entre 0,35 e aumentando até 0,50.



(a) USA (b) NSF

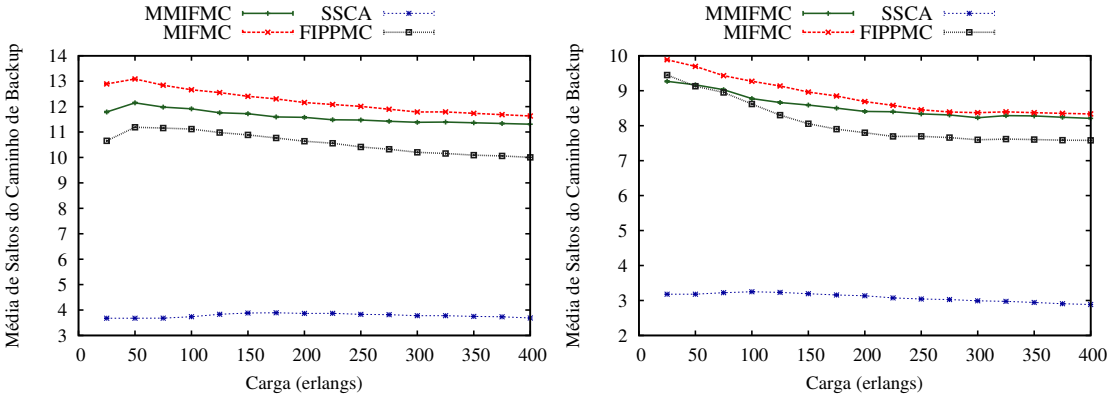
Figura 6. Número médio de saltos do caminho primário

As Figuras 6(a) e 6(b) mostram o número médio de saltos para o caminho primário para as topologia USA e NSF, respectivamente. Para ambas as topologias o algoritmo

MMIFMC possuem um número médio de salto constante, principalmente para cargas baixas.

Para a topologia USA (Figura 6(a)), nota-se que com o aumento da carga, os algoritmos MIFMC e FIPPMC diminuem o número médio de saltos no caminho primário, indicando a diminuição da disponibilidade dos enlaces que causam um maior bloqueio. Por sua vez, os algoritmos SSCA aumenta o número médio de saltos entre 100 e 250 erlangs demonstrando que a rede possui recursos ociosos, apesar do alto bloqueio.

Para a topologia NSF (Figura 6(b)), que possui baixa conectividade, similarmente a topologia USA, nota-se que com o aumento da carga, os algoritmos MIFMC e FIPPMC diminuem o número médio de saltos no caminho primário, indicando a diminuição da disponibilidade dos enlaces o que causa maior bloqueio. Por sua vez, os algoritmos SSCA aumenta o número médio de saltos entre 75 e 125 erlangs demonstrando que a rede possui recursos ociosos, apesar do alto bloqueio.



(a) USA (b) NSF

Figura 7. Número médio de saltos do caminho de backup

As Figuras 7(a) e 7(b) comparam o número médio de saltos para o caminho de backup nas topologia USA e NSF, respectivamente.

Para a topologia USA (Figura 7(a)), a necessidade de criação de anéis virtuais para proteção nos algoritmos FIPPMC, MIFMC e MMIFMC geram caminhos com até quatro vezes mais saltos que os algoritmos SSCA. Como p-cycles FIPP permitem ser compartilhados entre qualquer nó presente no p-cycle, o alto número de saltos não gera sobrecarga na rede e nem aumento do bloqueio da rede.

Para a topologia NSF (Figura 7(b)), similarmente a topologia USA, os algoritmos FIPPMC, MIFMC e MMIFMC geram caminhos com número de saltos três vezes maior do que os produzidos pelo algoritmo SSCA. A diferença no número de saltos nas topologias USA e NSF para os algoritmos FIPPMC, MIFMC e MMIFMC é devido à menor conectividade da rede. Em ambas as topologias, apesar dos algoritmos que utilizam *p-cycle* necessitarem de um caminho maior, a alta capacidade de compartilhamento consegue prover proteção para um maior número de caminhos, diferentemente das outras soluções, o que não aumenta a probabilidade de bloqueio.

Conclusão

Este artigo introduziu um algoritmo para suportar o estabelecimento de caminhos ópticos em redes ópticas elásticas com multiplexação espacial protegidas por p-cycle utilizando modulação adaptativa e interferência mínima. O algoritmo foi avaliado para diferentes topologias e cargas. O algoritmo desenvolvido produz menor probabilidade de bloqueio que os outros algoritmos existentes na literatura. Os resultados indicam que o algoritmo proposto pode fornecer proteção pré-configurada eficientemente para redes ópticas elásticas com multiplexação espacial. O algoritmo MMIFMC tem menor média de bloqueio em topologias com alta conectividade. O grau médio do nó na topologia de rede tem grande influência na relação de bloqueio de largura de banda e no comprimento de caminhos estabelecidos. A técnica p-cycle realiza restauração rápida para redes em malha. O uso de interferência mínima na criação do p-cycle, distribui melhor a carga na rede. Além disso, a utilização de modulação adaptativa, possibilita uma melhor utilização dos recursos da rede.

Agradecimento

O presente trabalho foi realizado com apoio do CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil, processo número 165446/2015-3 e INCT Fotonicom.

Referências

Asthana, R., Singh, Y., and Grover, W. (2010). p-cycles: An overview. *Communications Surveys Tutorials, IEEE*, 12(1):97 –111.

Chen, X., Zhu, S., Jiang, L., and Zhu, Z. (2015). On spectrum efficient failure-independent path protection p-cycle design in elastic optical networks. *Lightwave Technology Journal of*, 33(17):3719–3729.

Costa, L. R. and Drummond, A. C. (2016). New distance-adaptive modulation scheme for elastic optical networks. *IEEE Communications Letters*, PP(99):1–1.

Costa, L. R., Ramos, G. N., and Drummond, A. C. (2016). Leveraging adaptive modulation with multi-hop routing in elastic optical networks. *Computer Networks*, 105:124 – 137.

Essiambre, R., Kramer, G., Winzer, P., Foschini, G., and Goebel, B. (2010). Capacity limits of optical fiber networks. *Lightwave Technology, Journal of*, 28(4):662–701.

Figueiredo, G. B., da Fonseca, N. L., and Monteiro, J. A. (2006). A minimum interference routing algorithm with reduced computational complexity. *Computer Networks*, 50(11):1710 – 1732.

Figueiredo, G. B., da Fonseca, N. L. S., and Monteiro, J. A. S. (2004). A minimum interference routing algorithm. In *2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577)*, volume 4, pages 1942–1947 Vol.4.

Fujii, S., Hirota, Y., Tode, H., and Murakami, K. (2014). On-demand spectrum and core allocation for reducing crosstalk in multicore fibers in elastic optical networks. *Optical Communications and Networking, IEEE/OSA Journal of*, 6(12):1059–1071.

Jinno, M., Kozicki, B., Takara, H., Watanabe, A., Sone, Y., Tanaka, T., and Hirano, A. (2010). Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network [topics in optical communications]. *IEEE Communications Magazine*, 48(8):138–145.

Klonidis, D., Zakynthinos, P., and Tomkos, I. (2015). Opportunities and challenges in the network planning of spatially and spectrally elastic optical networks. In *Transparent Optical Networks (ICTON), 2015 17th International Conference on*, pages 1–4.

Kodian, A. and Grover, W. (Oct. 2005). Failure-independent path-protecting pcycles: efficient and simple fully preconnected optimal-path protection. *IEEE, J. Lightwave Technol.*, 23:3241–3259.

Moura, P. M. and Drummond, A. C. FlexGridSim: Flexible Grid Optical Network Simulator. http://www.lrc.ic.unicamp.br/FlexGridSim/.

Moura, P. M., Scaraficci, R. A., and d. Fonseca, N. L. S. (2015). Algorithm for energy efficient routing, modulation and spectrum assignment. In *2015 IEEE International Conference on Communications (ICC)*, pages 5961–5966.

Muhammad, A., Zervas, G., Simeonidou, D., and Forchheimer, R. (2014). Routing, spectrum and core allocation in flexgrid sdm networks with multi-core fibers. In *Optical Network Design and Modeling, 2014 International Conference on*, pages 192–197.

Oliveira, H. and Fonseca, N. (2016). Proteção de redes Ópticas elásticas baseada em agregação de tráfego, sobreposição de espectro e p-cycle fipp. In *SBRC 2016*, Salvador, Bahia.

Oliveira, H. M. N. S. and Da Fonseca, N. L. S. (2016a). Algorithm for protection of space division multiplexing elastic optical networks. In *Global Communications Conference (GLOBECOM), 2016 IEEE*. IEEE.

Oliveira, H. M. N. S. and Da Fonseca, N. L. S. (2016b). The minimum interference p-cycle algorithm for protection of space division multiplexing elastic optical networks. In *Latin-American Conference on Communications (Latincom), 2016 IEEE*. IEEE.

Tode, H. and Hirota, Y. (2014). Routing, spectrum and core assignment for space division multiplexing elastic optical networks. In *Telecommunications Network Strategy and Planning Symposium (Networks), 2014 16th International*, pages 1–7.

Tomkos, I., Zakynthinos, P., Klonidis, D., Marom, D., Sygletos, S., Ellis, A., Salvadori, E., Siracusa, D., Angelou, M., Papastergiou, G., Psaila, N., Ferran, J. F., Ben-Ezra, S., Jimenez, F., and Fernández-Palacios, J. P. (2013). Spatial-spectral flexible optical networking: enabling switching solutions for a simplified and efficient sdm network platform. *Proc. SPIE*, 9009.

Vizcaíno, J. L., Ye, Y., and Monroy, I. T. (2012). Energy efficiency analysis for flexible-grid ofdm-based optical networks. *Computer Networks*, 56(10):2400 – 2419. Green communication networks.

Zhang, Y., Yan, L., Wang, H., and Gu, W. (2012). Routing, wavelength and mode assignment algorithm for space division multiplexing transmission network. In *Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2012 Second International Conference on*, pages 1383–1385.

Proteção Dedicada para Redes Ópticas Elásticas Considerando Efeitos de Camada Física

Jurandir Lacerda Jr[1,2], Alexandre Fontinele[3], Divanilson Campelo[3], André Soares[2]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Piauí - IFPI
Corrente – PI – Brasil

[2]Departamento de Computação
Universidade Federal do Piauí - UFPI
Teresina – PI – Brasil

[3]Centro de Informática - CIn
Universidade Federal de Pernambuco - UFPE

jurandir.cavalcante@ifpi.edu.br

Abstract. *Elastic Optical Network is a promising technology to compose the Internet infrastructure. For this network it is necessary to provide mechanisms that guarantee its availability even after the occurrence of a failure. It is also important to provide quality of the optical signal, which tends to degrade by the effect of the physical layer impairments. This paper proposes three algorithms that use the dedicated path protection strategy: DP-SNR, DP-BSNR and DP-RQoTO. These algorithms take into account the effects of physical layer impairments in their choices of routes. An evaluation was made with dynamic traffic and requisitions with variable band demands. The three algorithms presented in this paper were compared with other literature proposals, obtaining a reduction in the blocking probability in the order of 90%. Among the proposed algorithms, the DP-RQoTO obtained a decrease in the blocking probability of at least 36% in relation to the DP-SNR and 34% in the DP-BSNR in the EON topology. In the USA topology, the minimum gain of DP-RQoTO was 22% in relation to DP-SNR and 20% in relation to DP-BSNR.*

Resumo. *A Rede Óptica Elástica é uma tecnologia promissora para compor a infraestrutura do núcleo da Internet. Para este tipo de rede é necessário prover mecanismos que garantam a sua disponibilidade mesmo após a ocorrência de uma falha. Também é importante garantir a qualidade do sinal óptico, que tende a se degradar pelos efeitos de camada física. Este trabalho propõe três algoritmos que utilizam a estratégia de proteção dedicada de caminho: DP-SNR, DP-BSNR e DP-RQoTO. Tais algoritmos levam em consideração os efeitos de camada física nas escolhas de suas rotas. Foi realizada uma avaliação de desempenho com tráfego dinâmico e requisições com demandas de banda variáveis. Os três algoritmos apresentados neste artigo foram comparados com outras propostas da literatura, obtendo uma diminuição da probabilidade de bloqueio na ordem de 90%. Entre os algoritmos propostos, o DP-RQoTO obteve uma diminuição na probabilidade de bloqueio de no mínimo 36% em relação o DP-SNR e de 34% para o DP-BSNR na topologia EON. Na topologia USA, o ganho mínimo do DP-RQoTO foi de 22% em relação ao DP-SNR e 20% em relação ao DP-BSNR.*

1. Introdução

O crescimento do número de aplicações que fazem uso da Internet vem exigindo maior eficiência das tecnologias adotadas na sua infraestrutura. Aplicações como *Stream* de audio ou vídeo inserem uma quantidade de dados substancial na rede. Desta forma, é necessário um esforço da comunidade científica para propor melhorias nas redes de transporte. A rede óptica é a principal tecnologia para suportar tal demanda [Chatterjee et al. 2015].

Recentemente a tecnologia *Orthogonal Frequency Division Multiplexing* (OFDM) vem sendo apontada como solução mais eficiente de multiplexação para redes ópticas de transporte. As redes ópticas de transporte que fazem uso da multiplexação OFDM são denominadas de *Spectrum-Sliced Elastic Optical Path Network* (SLICE) ou redes ópticas elásticas [Jinno et al. 2009]. A rede óptica elástica proporciona uma maior eficiência no uso dos recursos ópticos, se comparada as atuais redes que adotam a tecnologia *Wavelength Division Multiplexing* (WDM). Tal eficiência é alcançada devido a divisão do espectro ópticos em pequenos intervalos denominados *slots* [Jinno et al. 2009].

No contexto das redes ópticas elásticas, um dos principais desafios é o problema *Routing, Modulation Level, and Spectrum Allocation* (RMLSA) [Costa and Drummond 2016] [Christodoulopoulos et al. 2011]. Tal problema consiste em realizar o roteamento e a alocação de espectro (alocação de um conjunto de *slots*), além da escolha do formato de modulação de maneira adaptativa.

A tolerância à falhas é uma capacidade fundamental no âmbito das redes ópticas elásticas. Essa capacidade é também conhecida como sobrevivência. Portanto, as operadoras fazem uso de técnicas de sobrevivência na tentativa de garantir o funcionamento da rede mesmo após a ocorrência de uma eventual falha.

Outro desafio no planejamento e operação da rede óptica elástica é reduzir os impactos das imperfeições de camada física na qualidade do sinal óptico. Apesar dos diferentes estudos da sobrevivência em redes ópticas e dos efeitos de camada física, são poucos os trabalhos que avaliam as técnicas de sobrevivência considerando as limitações de camada física em redes ópticas elásticas. Conforme demonstrado em [Lacerda Jr et al. 2016a, Lacerda Jr et al. 2016b], o uso de técnicas de sobrevivência sem considerar os efeitos de camada física ocasiona uma alta probabilidade de bloqueio de circuitos. O objetivo deste artigo é propor e avaliar três algoritmos RMLSA com proteção dedicada de caminho ciente dos efeitos de camada física.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 apresenta os principais conceitos sobre as redes ópticas elásticas. A Seção 3 discute algumas técnicas de sobrevivência para este tipo de rede. A Seção 4 apresenta os problemas de camada física relevantes no contexto das redes ópticas elásticas. A Seção 5 apresenta os três algoritmos propostos neste trabalho. Já na Seção 6 é feita uma avaliação de desempenho dos algoritmos propostos e os comparam com outros dois algoritmos de sobrevivência. Por fim, na Seção 7 são apresentadas as conclusões deste trabalho.

2. Redes Ópticas Elásticas

A tecnologia OFDM potencializa a eficiência espectral das redes ópticas elásticas [Jinno et al. 2009]. Em uma rede de transporte OFDM, a divisão do espectro óptico é

feita em pequenos canais de comunicação, conhecidos como *slots* de frequência. Um conjunto destes *slots* é usado para o estabelecimento de um circuito óptico. Devido esta divisão de menor granularidade, os recursos são usados de forma mais eficientes, evitando desperdícios [Horota et al. 2014]. Este trabalho tem foco nas redes ópticas elásticas transparentes. Neste tipo de rede, todo o roteamento é feito no domínio óptico, dispensando o uso de conversores *Óptico-Eletro-Óptico* (OEO).

A quantidade de *slots* que cada circuito usa é definido pelo formato de modulação adotado. A modulação é a forma de representação das informações digitais do canal óptico em sinais que representam os *bits* [Tanenbaum and Wetherall 2011]. Em uma rede óptica elástica, para cada circuito deve ser escolhido o formato de modulação mais adequado. Alguns formatos de modulação tem a capacidade de transportar mais bits por símbolo, como por exemplo a 32QAM. Devido a sua densidade de informação, tais formatos de modulação estão mais suscetíveis a erros em rotas longas. Já os formatos de modulação mais simples, como a BPSK, são mais indicados para este tipo de rota. A Figura 1 exemplifica alguns formatos de modulação.

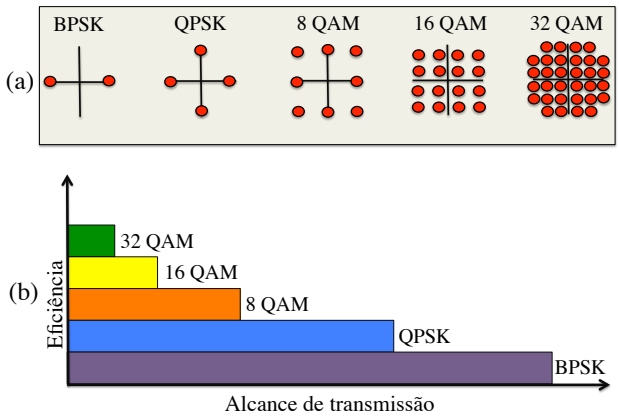


Figura 1. Exemplo de modulação.

A Figura 1 (a) mostra de forma simplista a quantidade de *bits* por símbolo que os formatos de modulação BPSK, QPSK, 8QAM, 16 QAM e 32QAM utilizam. A modulação 32QAM, apesar de ter uma maior eficiência espectral, está mais sujeita a erros durante a propagação do sinal ao longo da rota. O comportamento da relação entre eficiência óptica e alcance de transmissão pode ser exemplificada na Figura 1 (b).

Nas redes ópticas elásticas, o problema *Routing, Modulation Level, and Spectrum Allocation* RMLSA consiste em: (i) definir uma rota para um par de nós origem e destino; (ii) selecionar um formato de modulação que transporte mais *bits* por símbolo mas percorra toda a rota respeitando a qualidade do sinal óptico exigida pela rede; (iii) alocar uma faixa de espectro livre na rota definida para estabelecer o circuito óptico. A quantidade de *slots* é definida com base na largura de banda requisitada e na eficiência espectral do formato de modulação escolhido.

Para estabelecer um circuito na rede óptica elástica transparente, o algoritmo RMLSA deve obedecer duas restrições: (i) contiguidade espectral, que estabelece que os *slots* de um circuito devem ser adjacentes entre sí; (ii) continuidade espectral, que define

que tal conjunto de *slots* seja o mesmo em todos os enlaces da rota. Tais restrições devem ser respeitadas pelos algoritmos RMLSA, caso contrário o novo circuito não poderá ser estabelecido, ocasionando um bloqueio.

3. Sobrevivência em Redes Ópticas Elásticas

As redes ópticas elásticas permitem transportar grandes volumes de informações e fornecem serviços para aplicações que exigem rigorosos requisitos. Além da sensibilidade ao atraso e a interrupções de suas conexões, estas aplicações geralmente funcionam 24 horas por dia, exigindo uma alta disponibilidade de serviços da rede. A falha de um enlace da rede óptica representa a interrupção de todos os circuitos ópticos que utilizam tal enlace. Isto pode provocar a perda de uma grande quantidade de informações e a parada de serviços críticos. Diante disto, as redes ópticas elásticas precisam implementar mecanismos para garantir que suas conexões sejam recuperadas de maneira rápida e eficiente. Esta capacidade de continuar operando na eventualidade de ocorrência de falhas é conhecida como sobrevivência em redes ópticas.

Diferentes autores vem propondo e avaliando técnicas de sobrevivência em redes ópticas [Ruan and Zheng 2014, Shen et al. 2014, Wei et al. 2014, Amar et al. 2015, Chen et al. 2015, Wang et al. 2015, Shen et al. 2016]. Tais técnicas de sobrevivência podem ser classificadas em: proteção ou restauração.

A proteção é uma técnica proativa, que consiste na computação e reserva prévia de recursos redundantes antes da ocorrência de falhas. Tais recursos redundantes somente serão efetivamente utilizados para recuperar uma eventual falha. A proteção dedicada e a proteção compartilhada são as duas principais formas de prover a proteção. Na proteção dedicada, ao se alocar a rota primária, denominada rota de trabalho, também é alocada uma rota de *backup* de igual capacidade. Já na proteção compartilhada, a rota de *backup* é compartilhada entre outras requisições disjuntas entre si. Isso diminui o desperdício no uso dos recursos da rede.

A restauração é uma técnica reativa, que trata a falha apenas após o acontecimento da mesma. A restauração apesar de ter uma economia de recursos em relação a proteção, não consegue garantir que haverá recursos livres no momento da falha para alocar uma segunda rota. A Figura 2 mostra o funcionamento da proteção e da restauração.

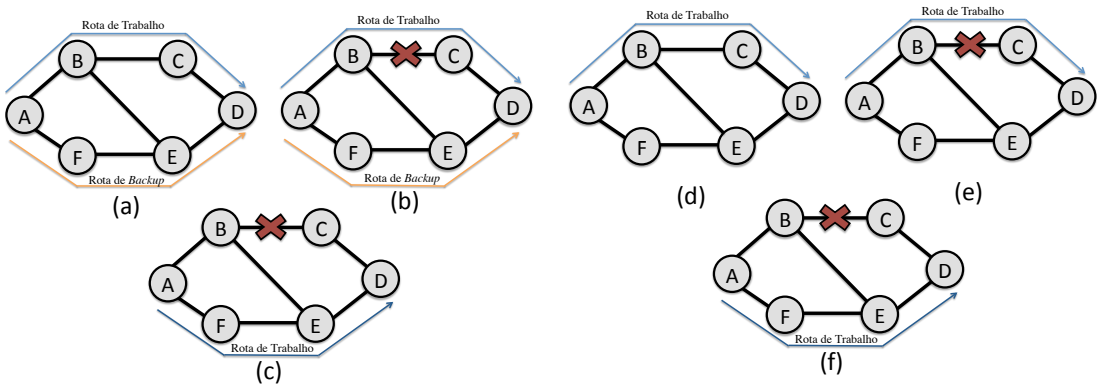


Figura 2. Exemplo de proteção dedicada e restauração dinâmica.

Na Figura 2 (a), (b) e (c) temos um exemplo de proteção dedicada, onde no momento de alocação do circuito (Figura 2(a)), são alocadas duas rotas: a rota de trabalho (A-B-C-D) e a rota de *backup* (A-F-E-D). A rota de trabalho é usada para o fluxo normal das informações e a rota de *backup* será usada caso algum enlace da rota de trabalho falhe. No momento exemplificado na Figura 2(b), ocorre uma falha no enlace B-C, que faz parte da rota de trabalho. Para a rede continuar operante, automaticamente o recurso alocado como *backup* passa a funcionar como a nova rota de trabalho, como mostrado na Figura 2(c). Já na restauração, exemplificada na Figura 2 (d), (e) e (f), no momento da alocação da rota de trabalho (Figura 2(d)), nenhuma rota de *backup* é alocada. No momento da falha, mostrado na Figura 2(e), o algoritmo de restauração irá entrar em ação e buscar uma rota alternativa, como mostra a Figura 2(f). Caso não exista recursos livres, a conexão será perdida. Devido a maior garantia de sobrevivência, a proteção é a classe de técnicas abordada por este trabalho. Em especifico, a proteção dedicada.

4. Efeitos de Camada Física

Naturalmente, ocorre uma degradação da qualidade do sinal óptico durante sua propagação. Isso ocorre devido aos efeitos de camada física, tanto nos dispositivos dos nós da rede, quanto nos próprios enlaces. A literatura classifica tais degradações em: efeitos lineares (*Linear Impairments* - LI) e efeitos não lineares (*Nonlinear Impairments* - NLI) [Rahbar 2012]. Os Efeitos Lineares são aqueles independentes da potência do sinal. A Dispersão Cromática (*Chromatic Dispersion* - CD), a Emissão Espontânea Amplificada (*Amplified Spontaneous Emission* - ASE) e a Atenuação da Fibra são efeitos desta categoria. Já os Efeitos Não Lineares são dependentes da potência dos sinais ópticos e podem causar interferências tanto no próprio circuito, como nos seus vizinhos. A Auto-Modulação de Fase (*Self-Phase Modulation* - SPM), a Modulação de Fase Cruzada (*Cross-Phase Modulation* - XPM) e a Mistura de Quatro Ondas (*Four-Wave Mixing* - FWM) são exemplos de efeitos não lineares.

Durante a transmissão de um sinal, quanto maior a distância percorrida deste na fibra, maior será a atenuação da potência do sinal. Isso ocorre de tal forma que gera uma necessidade em amplificar o sinal óptico para que este restabeleça sua potência e assim possa ser detectado no destino. Usualmente, a amplificação óptica é realizada pelo equipamento amplificador *Erbium Doped Fiber Amplifer* (EDFA). Amplificadores EDFA introduzem, naturalmente, o ruído ASE [Saradhi and Subramaniam 2009]. Este trabalho considera o ruído ASE.

Além disso, são considerados a ocorrência dos efeitos SPM, XPM e FWM. O SPM ocorre porque o índice de refração da fibra possui uma componente dependente da intensidade. Isso provoca um deslocamento da fase induzida que é proporcional à intensidade do pulso e faz com que diferentes partes do pulso sofram diferentes deslocamentos de fase [Ramaswami and Sivarajan 2009]. O XPM é o deslocamento de fase de um sinal causado pelas flutuações de intensidade de outros canais que compartilham a mesma fibra a diferentes frequências ópticas. Esse efeito pode ser reduzido aumentando o espaçamento entre os circuitos ou fazendo com que circuitos operem a taxas de bits diferentes [Saradhi and Subramaniam 2009]. Por fim, o FWM ocorre pela ação não linear entre três frequências ópticas que dão origem a uma quarta frequência. Caso a quarta frequência seja igual ou próxima a uma das outras três, esse sinal interferirá na largura de banda do circuito [Saradhi and Subramaniam 2009]. Considerar esses quatro efeitos está

em consonância com o que a literatura trata. Para este trabalho utilizaremos o modelo de camada física proposto em [Johannisson and Agrell 2014, Zhao et al. 2015].

Em uma rede óptica elástica, tais efeitos de camada física podem impactar na qualidade do sinal óptico, pois a sua taxa de erro de *bit* (*Bit Error Rate* - BER) pode se tornar intolerável. Neste sentido, se a BER chegar a níveis elevados, a qualidade de transmissão (*Quality of Transmission* - QoT) será impactada. Com isso, poderá gerar um bloqueio por QoT [Beyranvand and Salehi 2013]. Os receptores ópticos possuem uma curva de desempenho que associa a relação sinal ruído (*Signal to Noise Ratio* - SNR) diretamente com a BER, portanto, a SNR pode ser usada como critério de QoT de camada física de um circuito óptico. Detalhamentos sobre o cálculo de SNR adotado neste trabalho pode ser encontrado em [Johannisson and Agrell 2014, Zhao et al. 2015].

Caso os níveis de SNR não sejam adequados, a requisição pode ser bloqueada por QoTN (QoT no novo circuito) ou QoTO (QoT nos outros circuitos já estabelecidos) [Fontinele et al. 2016a]. O QoTN é o bloqueio sofrido caso a nova requisição não atinja os níveis adequados de QoT. Mesmo que uma nova requisição atinja tal requisito, ela ainda poderá sofrer bloqueio caso o estabelecimento da nova requisição impacte na QoT dos circuitos já estabelecidos, ocasionando assim o QoTO. Estes dois tipos de bloqueio são considerados na avaliação de desempenho deste trabalho.

5. Algoritmos Propostos

Neste artigo, são propostos três algoritmos RMLSA com proteção dedicada de caminho ciente dos efeitos de camada física: (i) *Dedicated Protection with evaluation of SNR* (DP-SNR); (ii) *Dedicated Protection with evaluation of the Best SNR* (DP-BSNR); e (iii) *Dedicated Protection with Reduction of QoTO* (DP-RQoTO).

Algoritmo 1 : DP-SNR

1: Computa K menores caminhos para par $(Origem, Destino)$
2: i = 0
3: Calcula a Modulação e faixa de $slots$ para a rota K_i
4: **if** existe rota disjunta para a rota K_i **then**
5: Calcula modulação e faixa de $slots$ para a rota disjunta da rota K_i
6: $Info_{trabalho}$ = Informações de rota, modulação, $slots$ referente a K_i
7: $Info_{backup}$ = Informações de rota, modulação, $slots$ referente a rota disjunta da rota K_i
8: **if** SNR de $Info_{trabalho}$ e $Info_{backup}$ estão de acordo com os respectivos limiares dos formatos de modulação **then**
9: **return** $Info_{trabalho}$ e $Info_{backup}$
10: **end if**
11: **end if**
12: i++
13: **if** $i = K$ **then**
14: **return** $null$
15: **else**
16: Volte para a linha 3
17: **end if**

O algoritmo DP-SNR é uma evolução direta da proteção dedicada de caminho. A proteção dedicada de caminho tradicional busca encontrar duas rotas disjuntas. A primeira destas rotas é escolhida como rota de trabalho e a segunda como rota de *backup*. Ao escolher tais rotas, o DP-SNR analisa se os níveis de SNR estão de acordo com o limiar preestabelecido para o formato de modulação selecionado. Caso o SNR de uma rota candidata não esteja de acordo, a mesma será descartada e o algoritmo irá testar a rota seguinte. O DP-SNR irá buscar rotas até achar um par de rotas (trabalho e *backup*) que tenham limiares de SNR em concordância com os exigidos pelos formatos de modulação selecionados para as respectivas rotas, como é exemplificado no Algoritmo 1.

Algoritmo 2 : DP-BSNR

1: $Info_{trabalho}$ = $null$
2: $Info_{backup}$ = $null$
3: Computa K menores caminhos para par $(Origem, Destino)$
4: **for** (int $i = 0$; $i < K$; i++) **do**
5: Calcula a Modulação e faixa de $slots$ para a rota K_i
6: **if** existe rota disjunta para a rota K_i **then**
7: Calcula modulação e faixa de $slots$ para a rota disjunta da rota K_i
8: **if** SNR da rota K_i e da rota disjunta da rota K_i estão de acordo com os respectivos limiares dos formatos de modulação e rota K_i tem valor de SNR melhor que $Info_{trabalho}$ e rota disjunta da rota K_i tem valor de SNR melhor que $Info_{backup}$ **then**
9: $Info_{trabalho}$ = Informações de rota, modulação, $slots$ referente a K_i
10: $Info_{backup}$ = Informações de rota, modulação, $slots$ referente a rota disjunta da rota K_i
11: **end if**
12: **end if**
13: **end for**
14: **return** $Info_{trabalho}$ e $Info_{backup}$

O algoritmo DP-BSNR é semelhante ao DP-SNR. Entretanto o DP-BSNR busca a melhor rota dentro de um conjunto de menores rotas, como mostrado no Algoritmo 2. O algoritmo irá selecionar um conjunto de k menores rotas de trabalho e, para cada uma destas rotas de trabalho, calcular uma rota disjunta (*backup*). O DP-BSNR irá percorrer todo o conjunto k e selecionar a rota com maior qualidade de sinal com base no valor de SNR. A rota de k selecionada será definida como rota de trabalho e sua respectiva rota disjunta será usada como *backup*.

O algoritmo DP-RQoTO é uma evolução do algoritmo KSP-RQoTO (*K-Shortest Path with Reduction of QoTO*), proposto em [Fontinele et al. 2016b]. O KSP-RQoTO é um algoritmo RMLSA ciente de camada física mas não provê sobrevivência, já o DP-RQoTO é capaz de garantir sobrevivência e tratar efeitos de camada física. O DP-RQoTO computa os k menores caminhos para cada par de nós origem e destino da rede em uma fase *off-line*. No momento que a rede está em operação, o algoritmo proposto busca alocar uma das k rotas alternativas de forma a minimizar o bloqueio do tipo QoTO (QoT inadequada para os outros circuitos ópticos já ativos na rede). Para isso, é exigido um limiar de SNR acima do limiar exigido pelo formato de modulação selecionado para um

determinada rota. Este valor de limiar adicional é chamado de σ. A adição do valor σ ajuda a selecionar um formato de modulação que torna o circuito em estabelecimento mais resistente a interferências de outros circuitos, que podem ser os já ativos ou que serão estabelecidos.

Algoritmo 3 : DP-RQoTO

1: $Info_{trabalho}$ = $null$
2: $Info_{backup}$ = $null$
3: M = Formatos de modulação em ordem crescente de eficiência espectral
4: Computa K menores caminhos para par $(Origem, Destino)$
5: **for** (int $i = 0$; $i < K$; i++) **do**
6: **for** (int $j = 0$; $j < M$; j++) **do**
7: **if** existe espectro livre e a QoT está aceitável para a rota K_i **then**
8: Computa o ΔSNR_{mod} e atribui como avaliação da modulação M_j
9: **end if**
10: **end for**
11: Seleciona o formato de modulação com $\Delta SNR_{mod} >= \sigma$ para a rota K_i
12: Calcula a faixa de $slots$ para a rota K_i
13: **if** existe rota disjunta para a rota K_i **then**
14: **for** (int $j = 0$; $j < M$; j++) **do**
15: **if** existe espectro livre e a QoT está aceitável para a rota disjunta da rota K_i **then**
16: Computa o ΔSNR_{mod} e atribuí como avaliação do formato de modulação M_j
17: **end if**
18: **end for**
19: **end if**
20: Seleciona o formato de modulação com $\Delta SNR_{mod} >= \sigma$ para a rota disjunta da rota K_i
21: Calcula a faixa de slots para a rota disjunta da rota K_i
22: **if** SNR da rota K_i e da rota disjunta da rota K_i estão de acordo com os respectivos limiares dos formatos de modulação e o índice de $slot$ para a rota K_i é menor que o índice de $slot$ da $Info_{trabalho}$ e o índice de $slot$ para a rota disjunta da rota K_i é menor que o índice de $slot$ da $Info_{backup}$ **then**
23: $Info_{trabalho}$ = Informações de rota, modulação, $slots$ referente a rota K_i
24: $Info_{backup}$ = Informações de rota, modulação, $slots$ referente a rota disjunta a K_i
25: **end if**
26: **end for**
27: **return** $Info_{trabalho}$ e $Info_{backup}$

Como apresentado no Algoritmo 3, o DP-RQoTO também tenta selecionar um formato de modulação para a rota de *backup* que permita estabelecer circuitos mais resistentes a interferências de outros circuitos. Para isso, o algoritmo busca selecionar formatos de modulação que possuem valor de ΔSNR que respeite a margem de segurança (σ). O valor do σ é definido previamente na fase de planejamento da rede, como é discutido

na Seção 6. Caso nenhum formato de modulação respeite o σ é selecionado o formato de modulação com a maior eficiência espectral que for possível estabelecer o circuito.

6. Avaliação de Desempenho

Para o estudo de avaliação de desempenho deste trabalho, foi utilizado o simulador *SLICE Network Simulator* (SNetS) [Santos 2015]. O SNetS é uma ferramenta de simulação desenvolvida para a avaliação de desempenho de redes ópticas elásticas OFDM [Santos 2015].

Para reforçar os resultados deste trabalho, também foi realizado um estudo de validação do simulador SNetS. Tal estudo é importante para comprovar que o simulador, de fato, reproduz os comportamentos e resultados que estão de acordo com outros trabalhos apresentados pela literatura. Para a realização deste estudo de validação, foram realizadas simulações de seis algoritmos distintos, apresentados em [Wan et al. 2012]. Para a realização de tais simulações, foram usados os mesmos parâmetros e a mesma modelagem definidos em [Wan et al. 2012]. Os resultados destas simulações apresentaram semelhanças aos obtidos em [Wan et al. 2012], como mostra a Figura 3.

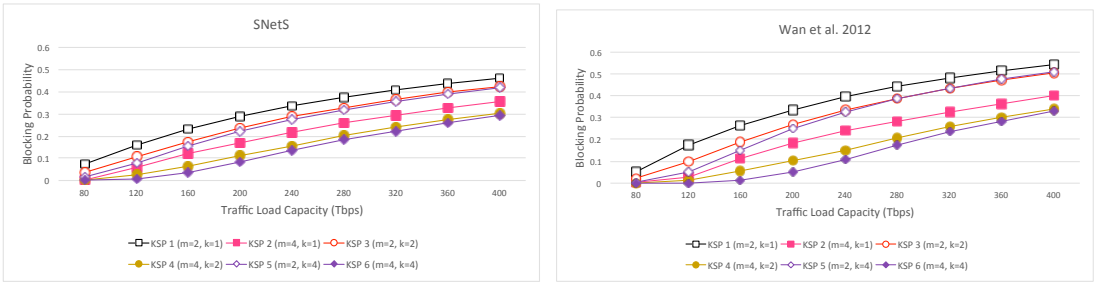


Figura 3. Probabilidade de bloqueio do KSP obtido pelo SNetS e em [Wan et al. 2012].

A partir dos gráficos da Figura 3, observa-se que o comportamento dos seis algoritmos, que são variações da técnica KSP (*K Shortest Path*), têm tanto o seu comportamento quanto os seus valores similares aos resultados obtidos em [Wan et al. 2012]. Esta semelhança é um forte indicio de que o simulador SNetS reproduz um resultado condizente com outros simuladores usados na literatura.

Para as simulações deste trabalho, foram geradas 100000 requisições de circuitos em cada simulação. A geração de requisições é um processo de Poisson com taxa média de λ e o tempo médio de retenção dos circuitos é distribuído exponencialmente com média $1/\mu$. A carga de tráfego é distruída uniformemente entre todos os pares de nós origem e destino. A carga em Erlangs pode ser definida por $\rho = \lambda/\mu$. Para cada simulação foram realizadas 10 replicações com diferentes sementes de geração de variável aleatória. Todos os resultados possuem nível de confiança de 95%. As topologias consideradas nas simulações são a EON e a USA (Figura 4). O valor apresentado em cada enlace da topologia indica a distância do enlace em km.

Os requisitos de taxas de bits para cada circuito requisitado variam uniformemente entre 10, 40, 80, 100, 160, 200 e 400 Gbps. Os formatos de modulação considerados neste estudo foram BPSK, QPSK, 8QAM, 16QAM, 32QAM e 64QAM, seus respectivos limiares de SNR são 6 dB, 9 dB, 12 dB, 15 dB, 18 dB e 21 dB [Beyranvand and Salehi 2013].

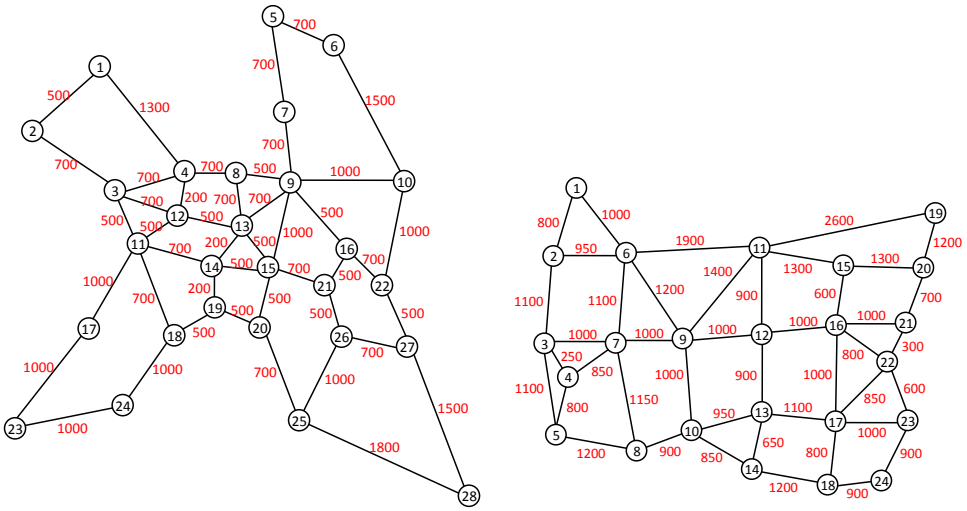


Figura 4. Topologia EON e USA.

Os respectivos níveis dos formatos de modulação são 2, 3, 4, 5, 6 e 7. Para se calcular a largura de banda de uma requisição i para uma dada taxa de bits B_n, com nível de modulação L_m e sobrecarga de FEC (*Forward Error Correction*) F, é utilizada a Equação 1 [Gao et al. 2014]:

$$B_i = \frac{1.1 B_n \left(1 + F\right)}{2 \log_2 L_m}. \tag{1}$$

Em seguida, encontra-se um número inteiro de slots de frequência que cubra a largura de banda requisitada acrescentada da banda de guarda. Neste estudo, foi considerada uma FEC de 7%, que corresponde a um limiar de BER de $3,8 \times 10^{-3}$ [Gao et al. 2014].

Todos os enlaces da rede são bidirecionais e possuem largura de banda do espectro dividida em 400 slots de frequência. Um slot de frequência possui largura de banda de 12,5 GHz e a banda de guarda possui largura de banda de 6,25 GHz [Gao et al. 2014]. Os ganhos dos amplificadores são ajustados para compensar as perdas dos dispositivos e da fibra. Outros parâmetros utilizados nas simulações estão listados na Tabela 1 [Beyranvand and Salehi 2013, Zhao et al. 2015].

Tabela 1. Parâmetros de camada física utilizados nas simulações.

Descrição	Valor
Densidade espectral de potência do sinal	-17 dBm/GHz
Atenuaçao da fibra (α)	0,2 dB/km
Parâmetro de dispersão da fibra (β_2)	16 ps^2/km
Coeficiente não linear da fibra (γ)	1,3 (Wkm)$^{-1}$
Tamanho de um span (L_s)	100 km
Figura de ruído do amplificador (NF)	6 dB
Potência de saturação do amplificador (P_{SAT})	16 dBm

Neste cenário, os algoritmos DP-SNR, DP-BSNR e DP-RQoTO foram comparados com os algoritmos de Proteção Dedicada de Caminho (*Dedicated Protection* -

DP) e o algoritmo *Survivable Multipath Routing and Spectrum Allocation* (SM-RSA) [Ruan and Zheng 2014]. O algoritmo SM-RSA pode ser configurado para trabalhar com um nível de sobrevivência variado. Para este trabalho, o SM-RSA foi analisado com exigências de sobrevivência de 100%, para que esteja de acordo com os demais algoritmos. Também foi realizado um estudo para decidir qual o valor de σ ideal para o algoritmo DP-RQoTO. A partir deste estudo, foi definido que o valor ideal para este cenário é $\sigma = 0.3$. Por questões de adequação ao limite de páginas, a análise do σ não será mostrada neste artigo. A Figura 5 apresenta os resultados, em termos de probabilidade de bloqueio, para a topologia EON.

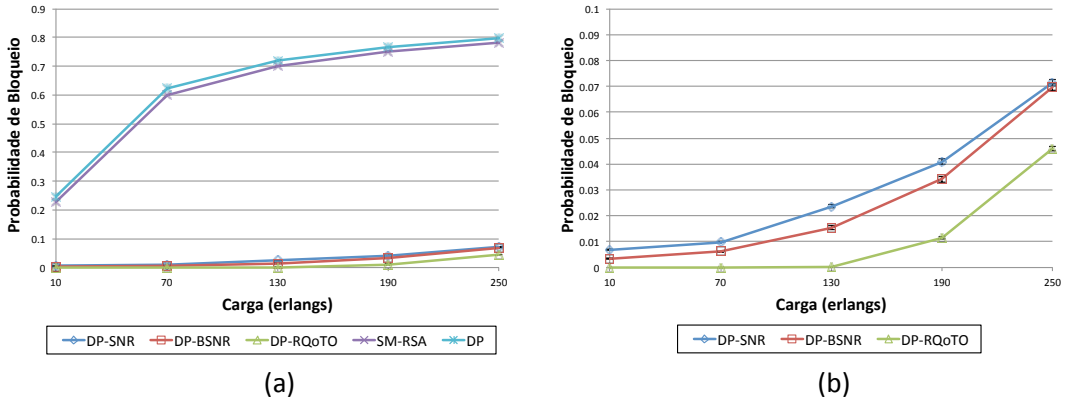


Figura 5. Probabilidade de bloqueio para a topologia EON.

A Figura 5(a) apresenta a probabilidade de bloqueio geral dos cinco algoritmos na topologia EON. É perceptível a melhora que os três algoritmos propostos obtiveram em relação os concorrentes. Esta melhora é devido ao fato deste cenário levar em consideração os efeitos de camada física e os respectivos bloqueios por ausência de QoT. Visto que comumente as técnicas de sobrevivência não levam em consideração esta problemática, os bloqueios por QoTN e QoTO apresentaram um impacto significativo nos algoritmos DP e SM-RSA, como apresentado em [Lacerda Jr et al. 2016b]. Para os algoritmos DP e SM-RSA, a partir de 130 *erlangs* de carga na rede, a probabilidade de bloqueio ultrapassou 70%. Neste mesmo cenário, os algoritmos DP-SNR, DP-BSNR e DP-RQoTO obtiveram probabilidade de bloqueio abaixo de 8%. Para melhor visualização da diferença de desempenho das estratégias propostas, a Figura 5(b) apresenta apenas estes algoritmos. O DP-RQoTO obteve o melhor resultado dentre todos os algoritmos analisados, com 4,6% de probabilidade de bloqueio no maior ponto de carga (250 *erlangs*), contra 7,1% do DP-SNR e 7,0% do DP-BSNR. O que representa uma diminuição da probabilidade de bloqueio de 34% (DP-BSNR) e 36% (DP-SNR). Os resultados são similares para a topologia USA, apresentada na Figura 6.

A Figura 6(a) apresenta a probabilidade de bloqueio geral dos algoritmos na topologia USA. Novamente, as estratégias propostas obtiveram um resultado superior quando comparadas aos algoritmos de proteção dedicada de caminho tradicional (DP) e o algoritmo SM-RSA. Para o maior ponto de carga (270 *erlangs*) a probabilidade de bloqueio do SM-RSA aproximou-se de 80% e a do DP ficou acima de 80%, enquanto os algoritmos propostos ficaram abaixo de 9% de bloqueio. A Figura 6(b) apresenta os resultados dos algoritmos propostos de forma isolada, para melhor visualização. A estratégia DP-

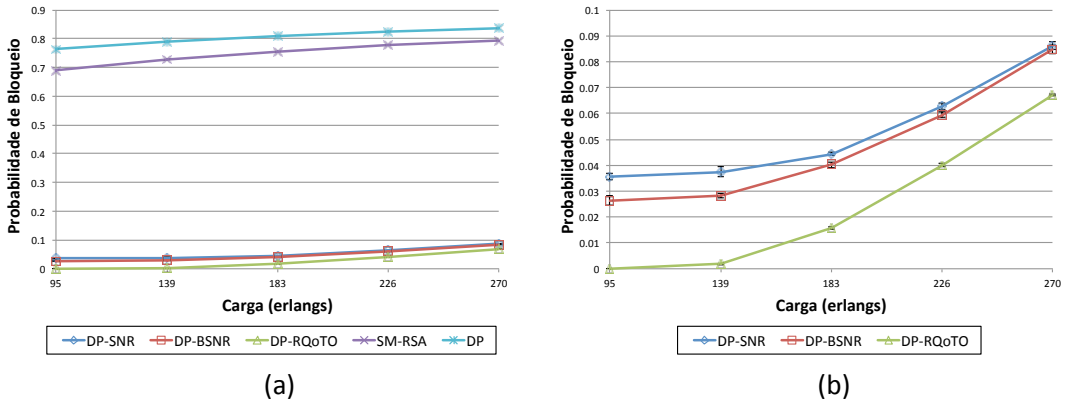


Figura 6. Probabilidade de bloqueio para a topologia USA.

RQoTO obteve novamente o melhor desempenho, com um bloqueio máximo de 6,7%. O maior bloqueio apresentado para o algoritmo DP foi 8,6%, para o DP-BSNR este valor foi de 8,4%. Em 270 *erlangs*, a diminuição da probabilidade de bloqueio do DP-RQoTO em relação ao DP-SNR e DP-BSNR foi de 22% e 20% respectivamente. Um ganho de desempenho maior do DP-RQoTO pode ser observado nos dois primeiros pontos de cargas (95 e 139 *erlangs*), onde a diminuição da probabilidade de bloqueio em relação os outros dois algoritmos ultrapassou 90%.

O ganho do algoritmo DP-RQoTO em relação aos demais está estritamente relacionado com a sua proposta em reduzir o bloqueio por QoTO. Ao escolher rotas levando em consideração o valor de σ, essas rotas terão uma qualidade de sinal bem acima do exigido. Desta forma, os circuitos estabelecidos serão mais robustos e resistentes à alocação de novos circuitos. Com isso, circuitos em estabelecimento irão sofrer menos bloqueio por QoTO, impactando na probabilidade de bloqueio geral da rede. Isso aponta para a importância de criar soluções de sobrevivência ciente de camada física, que a diminuam o bloqueio por QoT.

7. Conclusão

O objetivo deste artigo é a proposta de três algoritmos RMLSA com proteção dedicada de caminho e que levem em consideração as imperfeições da camada física. Foram propostos os algoritmos DP-SNR, DP-BSNR e DP-RQoTO e todos obtiveram um bom resultado, em termos de probabilidade de bloqueio, quando comparados a proteção dedicada de caminho tradicional e ao algoritmo SM-RSA. Isso se deve ao fato de que a maioria dos algoritmos de sobrevivência não considerarem os efeitos de camada física e consequentemente os bloqueio por baixa qualidade de sinal.

O Algoritmo DP-RQoTO obteve o melhor resultado tanto na topologia EON quanto na USA. Tal algoritmo usa um valor de σ predefinido para escolher rotas mais robustas, a fim de amenizar o impacto do estabelecimento de novos circuitos na qualidade dos outros circuitos já estabelecidos. Desta forma, o algoritmo diminui o bloqueio por QoTO e também por QoTN.

Para trabalhos futuros pretende-se melhorar a função de escolha do par de rotas de trabalho e *backup* do algoritmo DP-RQoTO. Uma otimização nesta escolha pode diminuir ainda mais a probabilidade de bloqueio. Também pretende-se adaptar o DP-RQoTO

para compartilhar rotas de *backup*, com o intuito de economizar recursos e consequentemente diminuir a probabilidade de bloqueio. Busca-se também o amadurecimento de novas técnicas para este cenário, pois o mesmo apresenta uma maior similaridade com um sistema real, ao levar em consideração tanto as imperfeições de camada física quanto a possibilidade de ocorrência de falhas na rede.

Referências

Amar, D., Rouzic, E. L., Brochier, N., and Lepers, C. (2015). Multilayer restoration in elastic optical networks. In *2015 International Conference on Optical Network Design and Modeling (ONDM)*, pages 239–244.

Beyranvand, H. and Salehi, J. (2013). A quality-of-transmission aware dynamic routing and spectrum assignment scheme for future elastic optical networks. *Journal of Lightwave Technology*, 31(18):3043–3054.

Chatterjee, B., Sarma, N., and Oki, E. (2015). Routing and spectrum allocation in elastic optical networks: A tutorial. *IEEE Communications Surveys Tutorials*, 17(3):1776–1800.

Chen, X., Zhu, S., Chen, D., Hu, S., Li, C., and Zhu, Z. (2015). On efficient protection design for dynamic multipath provisioning in elastic optical networks. In *2015 International Conference on Optical Network Design and Modeling (ONDM)*, pages 251–256.

Christodoulopoulos, K., Tomkos, I., and Varvarigos, E. A. (2011). Elastic bandwidth allocation in flexible ofdm-based optical networks. *Journal of Lightwave Technology*, 29(9):1354–1366.

Costa, L. R. and Drummond, A. C. (2016). Novo algoritmo rmlsa com roteamento multihop em redes Ópticas elásticas. In *XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Sistribuídos SBRC 2016*.

Fontinele, A., Santos, I., Durães, G., and Soares, A. (2016a). Achievement of fair and efficient regenerator allocations in translucent optical networks using the novel regenerator assignment algorithm. *Optical Switching and Networking*, 19, Part 1:22 – 39.

Fontinele, A., Santos, I., Neto, J., Campelo, D., and Soares, A. (2016b). Um novo algoritmo rsa ciente de imperfeições de camada física para redes Ópticas elásticas. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*.

Gao, G., Zhang, J., Wang, L., Gu, W., and Ji, Y. (2014). Influence of physical layer configuration on performance of elastic optical ofdm networks. *IEEE Communications Letters*, 18(4):672–675.

Horota, A., Figueiredo, G., and Fonseca, N. (2014). Algoritmo de roteamento e atribuição de espectro com minimização de fragmentação em redes ópticas elásticas. *XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Jinno, M., Takara, H., Kozicki, B., Tsukishima, Y., Sone, Y., and Matsuoka, S. (2009). Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies. *IEEE Communications Magazine*, 47(11):66–73.

Johannisson, P. and Agrell, E. (2014). Modeling of nonlinear signal distortion in fiber-optic networks. *Journal of Lightwave Technology*, 32(23):4544–4552.

Lacerda Jr, J., Fontinele, A., Moura, I., and Soares, A. (2016a). Avaliação de desempenho de algoritmos rsa para redes Ópticas elásticas com tolerância a falhas em cenário com imperfeições de camada física. In *XVII Workshop de Testes e Tolerância a Falhas (WTF) - (SBRC)*.

Lacerda Jr, J., Fontinele, A., Moura, I., and Soares, A. (2016b). The impact of physical layer impairment in survivability algorithms of elastic optical networks. In *XLII Latin American Computing Conference (CLEI)*.

Rahbar, A. G. (2012). Review of dynamic impairment-aware routing and wavelength assignment techniques in all-optical wavelength-routed networks. *IEEE Communications Surveys Tutorials*, 14(4):1065–1089.

Ramaswami, R. and Sivarajan, K. N. (2009). *Optical Network - A Practical Perspective*. Morgan Kaufmann Publishers, 3th edition.

Ruan, L. and Zheng, Y. (2014). Dynamic survivable multipath routing and spectrum allocation in ofdm-based flexible optical networks. *IEEE/OSA Journal of Optical Communications and Networking*, 6(1):77–85.

Santos, I. (2015). *Alocação de Recursos para o Estabelecimento de Circuitos em Redes Ópticas WDM e OFDM*. Universidade Federal do Piauí, Teresina.

Saradhi, C. V. and Subramaniam, S. (2009). Physical layer impairment aware routing (pliar) in wdm optical networks: issues and challenges. *IEEE Communications Surveys Tutorials*, 11(4):109–130.

Shen, G., Guo, H., and Bose, S. K. (2016). Survivable elastic optical networks: survey and perspective (invited). *Photonic Network Communications*, 31(1):71–87.

Shen, G., Wei, Y., and Bose, S. K. (2014). Optimal design for shared backup path protected elastic optical networks under single-link failure. *IEEE/OSA Journal of Optical Communications and Networking*, 6(7):649–659.

Tanenbaum, A. S. and Wetherall, D. J. (2011). *Redes de Computadores*. Pearson Education - Br, 5th edition.

Wan, X., Hua, N., and Zheng, X. (2012). Dynamic routing and spectrum assignment in spectrum-flexible transparent optical networks. *IEEE/OSA Journal of Optical Communications and Networking*, 4(8):603–613.

Wang, C., Shen, G., Chen, B., and Peng, L. (2015). Protection path-based hitless spectrum defragmentation in elastic optical networks: Shared backup path protection. In *2015 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3.

Wei, Y., Shen, G., and Bose, S. K. (2014). Span-restorable elastic optical networks under different spectrum conversion capabilities. *IEEE Transactions on Reliability*, 63(2):401–411.

Zhao, J., Wymeersch, H., and Agrell, E. (2015). Nonlinear impairment-aware static resource allocation in elastic optical networks. *Journal of Lightwave Technology*, 33(22):4554–4564.

Trilha Principal do SBRC 2017
Sessão Técnica 21
Redes Sociais e Aplicações de E-commerce

Um Método de Detecção de Regiões Funcionais Utilizando Dados de Redes Sociais

Alice A. F. Menezes[1], Josiel W. V. Santos[1], Bruno Á. Souza[1], Thais G. Almeida[1], Fabíola G. Nakamura[1], Eduardo F. Nakamura[1], Carlos M. S. Figueiredo[2]

[1]Universidade Federal do Amazonas (UFAM) – Manaus, AM – Brasil

[2]Universidade do Estado do Amazonas (UEA) – Manaus, AM – Brasil

{alice.menezes, jwvs, bruno.abia, tga, fabiola, nakamura}@icomp.ufam.edu.br,

cfigueiredo@uea.edu.br

Abstract. *Social Sensing is an emergent research area due to the amount of useful information that are provided by users of social networks. Several works in literature present studies with applications in natural disaster detection, traffic monitoring or analyzing dynamic of cities. In this way, this work presents a method to detect functional regions using data from location-based social networks. A dataset of 157,054 check-ins from Foursquare was used to characterize subregions in an urban area, determining functions for them. Comparing the results with government data, we verified that an accuracy of 86% was obtained for the proposed method. In addition, we also present two study cases for it: one related to traffic occurrences, and other related to urban area dynamics.*

Resumo. *O Sensoriamento Social é uma área de pesquisa emergente devido à quantidade de informações úteis que são fornecidas pelos usuários de redes sociais. Vários trabalhos na literatura apresentam estudos com aplicações em detecção de desastres naturais, monitoramento de tráfego ou análise da dinâmica de cidades. Nesta direção, este trabalho apresenta um método para a detecção de regiões funcionais utilizando dados de redes sociais baseadas em localização. Foi utilizado um conjunto de dados de 157.054 check-ins do Foursquare, com o objetivo de caracterizar subregiões em uma área urbana, determinando funções para as mesmas. Comparando os resultados com dados governamentais, verificamos que foi obtida uma acurácia de 86% para o método proposto. Além disso, também apresentamos dois estudos de caso para o mesmo: um relacionado à análise de ocorrências de trânsito, e outro relacionado à dinâmica das áreas urbanas.*

1. Introdução

Celulares e redes sociais, originalmente concebidos como ferramentas de comunicação, estão sendo cada vez mais utilizados como ferramentas inovadoras de coleta de dados para aplicações de monitoramento social e urbano [Steenbruggen et al. 2015]. Celulares, além de possuírem sensores como GPS, acelerômetro, microfone, câmera e giroscópio, estão cada vez mais conectados e seus usuários compartilhando dados voluntariamente em redes sociais. Tal fenômeno possibilitou o surgimento de métodos que permitem a observação do comportamento de cidades e seus ambientes de forma dinâmica e automatizada, o que vem sendo chamado de Sensoriamento Social [Aggarwal and Abdelzaher 2013].

Aliado a técnicas de Mineração de Dados, o Sensoriamento Social pode ser utilizado como uma fonte de informação útil sobre diversos aspectos do nosso cotidiano. Desta forma, aplicações interessantes vêm surgindo, como estudos de comportamentos epidemiológicos [Madan et al. 2010], de monitoramento socioeconômico [Liu et al. 2015], de comportamento social [Moturu et al. 2011] e de análise da dinâmica das cidades [Silva et al. 2012]. Tais estudos podem contribuir significativamente para auxiliar a tomada de decisões sobre planejamento urbano, tornando as cidades mais inteligentes e, consequentemente, melhorando a qualidade de vida de seus habitantes.

Dentre os trabalhos da literatura, há vários que aplicam sensoriamento social a partir do Twitter[1] e do Foursquare[2] devido a sua popularidade [Sakaki et al. 2010, Aiello et al. 2013, Silva et al. 2013]. Em particular, o Foursquare é uma rede social baseada em localização [Traynor and Curran 2012] que permite que seus usuários realizem *check-ins* e expressem suas opiniões, fazendo um breve resumo das experiências vividas no local por meio de comentários [Almeida et al. 2016, Menezes et al. 2016]. Isto possibilita estudos como a análise de mobilidade e detecção de regiões funcionais, alvo do estudo deste artigo.

Regiões funcionais são áreas, em um perímetro urbano, nas quais alguma função ou característica é associada [Cranshaw et al. 2012]. Esta função pode estar relacionada a aspectos econômicos, sociais, de mobilidade e etc. Como exemplos de função, podemos citar se uma área é residencial, relacionada a comércio ou diversão noturna. A verificação da distribuição de diferentes regiões funcionais é um importante fator em estudos urbanos, como para o planejamento do crescimento urbano, mas que antes da ascenção das redes sociais consistia na realização de custosos censos, que poderiam levar alguns anos para serem atualizados [Zhi et al. 2016]. A detecção de regiões funcionais utilizando dados sociais permite um monitoramento dinâmico e de baixo custo, o que facilita a tomada de decisões e execução de ações preventivas de acordo com o objeto de estudo.

Neste contexto, este trabalho apresenta um método de detecção de regiões funcionais que utiliza dados do Foursquare. O método associa dados sociais com regiões em uma determinada área urbana. Assim, o cotidiano das pessoas que habitam a área irá definir as funções das regiões. Como resultado, obtivemos uma acurácia de 86% usando como alvo a cidade de Nova Iorque, cidade que possui dados oficiais de planejamento urbano disponíveis. Além disso, este trabalho também apresenta como contribuição duas aplicações de estudo de caso: uma para auxiliar na análise de acidentes de trânsito utilizando regiões funcionais, tornando possível o melhor direcionamento de ações para a prevenção destes, e outra para o monitoramento da dinâmica urbana, que permite observar como a cidade evolui e, consequentemente, como planjejar seu futuro.

As próximas seções deste trabalho estão dividas da seguinte forma: na seção 2 apresentamos os trabalhos relacionados, na seção 3 apresentamos o método proposto. Na seção 4, apresentamos a base de dados utilizada, os experimentos realizados e o resultados obtidos. Na seção 5, apresentamos duas aplicações que utilizam regiões funcionais. Por fim, na seção 6, apresentamos as conclusões e as direções futuras para este trabalho.

[1] `http://twitter.com/`
[2] `https://pt.foursquare.com/`

2. Trabalhos Relacionados

Os dados de redes sociais tornaram possível a análise da dinâmica das cidades, para a caracterização de usuários e regiões. Neste sentido, no trabalho de Noulas et al. [Noulas et al. 2011a], é apresentado o primeiro estudo sobre comportamento humano com dados provenientes do Foursquare. Os autores coletaram aproximadamente 12.000.00 *check-ins* de usuários da rede social. É apresentado uma análise da dinâmica geo-temporal de atividades coletivas de usuários, a fim de demonstrar como *check-ins* provêem meios para descoberta de padrões de comportamento humano diário e semanal, propriedades urbanas de vizinhanças e transições recorrentes entre diferentes atividades.

Em outro trabalho, Noulas et al. [Noulas et al. 2011b] demonstra como informações semânticas sobre localidades e atividades sociais podem ser exploradas no contexto de aplicações móveis e pesquisa científica. Os autores propõe o uso da categoria de lugares para criação de ”impressões digitais”de usuários e localidades. Desta forma, pessoas podem ser caracterizadas a partir dos tipos de lugares que elas visitam, enquanto localidades podem ser modeladas de acordo com seus locais. Para tanto, os autores utilizam um algoritmo de agrupamento espectral [Shi and Malik 1997] para agrupar localidades e usuários.

No trabalho de Cranshaw et al. [Cranshaw et al. 2012], os autores apresentam uma nova metodologia para o estudo da dinâmica, estrutura e caraterísticas de uma cidade em larga escala. Para isso, eles utilizaram uma aborgagem na qual, a partir de uma quantidade massiva de dados geolocalizados provenientes do Foursquare, foi desenvolvido um algoritmo que mapeia áreas geográficas distintas de uma cidade fornecendo um visão em tempo real de qualquer mundaça no padrão de atividade da população. Tal algoritmo utiliza técnicas de agrupamento espectral e leva em consideração tanto a proximidade espacial dos usuários, quanto a proximidade social.

No trabalho de Silva et al. [Silva et al. 2012], os autores introduzem um nova técnica de visualização de dados, chamada *City Image*, cujo objetivo consiste em auxiliar o entendimento sobre a dinâmica de cidades. Para validar sua técnica, os autores utilizaram como base de dados 4,7 milhões de *check-ins* do Foursquare de usuários de oito cidades diferentes. O resultado de tal técnica é uma matriz quadrada que sumariza a dinâmica de cidades. Tal matriz é gerada com base em grafos de transição, modelados a partir da mobilidade de usuários de acordo com as categorias de localidades da rede social Foursquare.

Em Zang et al. [Zhang et al. 2013], os autores argumentam que a informação temporal referente à mobilidade humana é de extrema importância para análises urbanas, e portanto, deve ser considerada em conjunto com informações espaciais. Os autores utilizaram dados do Foursquare, provenientes das cidades de Nova Iorque e São Franciso, para analizar a influência da dinâmica temporal de atividades no estudo de mobilidade humana.

Em comparação com este trabalho, os trabalhos citados anteriormente não apresentam um estudo que relaciona as regiões funcionais e as atividades de seus habitantes, apesar de também utilizarem dados socias.

No artigo de Zhi et al. [Zhi et al. 2016], os autores introduziram um novo modelo para detecar regiões funcionais baseado no modelo LRA (*Low Rank Approximation*). Para

tanto, utilizaram uma base de dados de aproximadamente 15 milhões de *check-ins* provenientes da cidade de Shangai. Foram identificadas estruturas espaço-temporais latentes, que quando analisadas reveleram uma série de associações entre as atividades espaciais e temporais de cidadãos. Por meio do algoritmo K-means, os autores obtiveram 5 tipos de clusters que estavam diretamente relacionados com a combinação temporal de atividades dos cidadãos. Desta forma, os autores destacaram a correlação direta entre grupos de regiões e diferentes atividades durantes períodos variados no decorrer no dia.

O trabalho citado determina funções para as regiões diante 5 aspectos, enquanto que este trabalho baseia-se nas categorias do Foursquare. Além disso, nosso método permite que as funções das regiões possam ser analisadas tanto em conjunto quanto de forma individual. Por fim, ressaltamos que este trabalho apresenta possíveis aplicações para a utilização das regiões funcionais.

Existem ainda trabalhos utilizam dados de dispositivos móveis para determinar a característica das regiões. Este processo é mais custoso e a mobilidade dos usuários não está explicitamente ligada a uma atividade. No trabalho de Xiang et al. [Xiang et al. 2013], os autores realizam partições de regiões por meio da análise da trajetória de cidadãos utilizando um modelo de tópicos latentes (LDA), no qual são consideradas distribuições de probabilidades ao particionar regiões. A base de dados utiliza contém 500 milhões de chamadas de celulares provenientes de cidadãos da China. Ao invés de utilizar algoritmos de agrupamento baseados em similaridade espacial, os autores consideram informações temporais a fim de determinar regiões funcionais.

O trabalho aqui apresentado diferencia-se dos anteriores por levar em consideração o cotidiano dos habitantes da região estudada, mostrando que as regiões funcionais podem alterar sua função ao longo do tempo, permitindo que os aspectos da dinâmica das cidades seja analisado de forma espaço-temporal.

3. Detecção de Regiões Funcionais

Neste artigo, propomos um método para a detecção de regiões funcionais em uma área urbana da seguinte forma (Figura 1):

1. Definição de dados sociais geolocalizados, com funções relacionadas a eles;
2. Definição de uma grade com um limite definido sobre uma determinada área urbana;
3. Associação de dados sociais com as células da grade, de forma que cada célula possua diversas funções;
4. Ranqueamento das funções em cada célula, verificando a categoria do Foursquare com a quantidade predominante de *check-ins* e fazendo com que a mesma seja determinada a função principal da região;
5. Verificação e agrupamento de regiões adjacentes que possuam a mesma função principal.

Como resultado, obtemos regiões compostas por polígonos, formados por coordenadas, e funções extraídas de dados sociais compartilhados por usuários da área urbana estudada. As funções das regiões definidas podem variar em dias e meses por exemplo, devido às mudanças dos dados das redes sociais ao longo do tempo. Desta forma, podemos dizer que as regiões funcionais são estabelecidas através de atividades humanas.

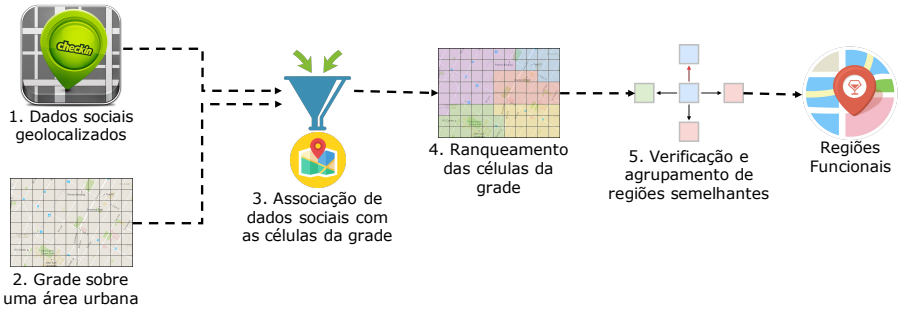


Figura 1. Método proposto.

De acordo com o método proposto, definimos que as regiões funcionais possuem as seguintes variáveis:

- Áreas formadas por uma lista de polígonos P, de forma que $P = \{p_1, p_2, p_3, ..., p_i\}$ e i é o total de células da grade. O valor de i é definido no passo 2, pois ao estabelecermos a área em que a grade será formada, informamos quatro pontos, NLa, SLa, LLo e OLo referentes as latitudes norte e sul e as longitudes leste e oeste, respectivamente. Em seguida, esta área é dividida em subregiões pertencentes a lista P.
- Funções predominantes formadas por dados sociais geolocalizados F, em que $F = \{f_1, f_2, f_3, ..., f_j\}$ e j é a quantidade de funções referentes a cada polígono pertencente a lista P. No passo 3, diversos dados sociais são associados a cada célula da grade, fazendo com que as mesmas possuam diversas funções. Desta forma, as funções predominantes são determinadas após o ranqueamento das funções das células da grade, no passo 4, em que apenas a função de maior ocorrência permanece.
- Intervalos de tempo T, de forma que $T = \{t_1, t_2, t_3, ..., t_k\}$ e k é a quantidade de intervalos de tempo. Estes intervalos de tempo podem ser estabelecidos em horas, dias, semanas, meses e etc.

Assim, antes do passo 5, possuímos uma matriz que associa P a F para cada intervalo de tempo em T, de forma que $M = \{PF_{t_1}, PF_{t_2}, PF_{t_3}, ..., PF_{t_k}\}$.

Por fim, para os intervalos de tempo em T, verificamos para cada região em P se as regiões adjacentes possuem a mesma função contida em F, de forma que as regiões semelhantes sejam interpoladas formando as matrizes contidas em M', de forma que $M' = \{P'F'_{t_1}, P'F'_{t_2}, P'F'_{t_3}, ..., P'F'_{t_k}\}$, P' possui tamanho $i' < i$ e F' possui tamanho $j' < j$.

O método proposto também permite que possamos analisar as regiões funcionais tanto em conjunto (todos os valores em F) quanto de forma individual. Assim, também é possível formar os polígonos em P, e posteriormente em P', baseando-se apenas em uma função de estudo.

4. Experimentos e Resultados

Neste trabalho, definimos que as funções contidas em F são referentes às categorias do Foursquare (Tabela 1). Escolhemos esta rede social pois a mesma é baseada em

localização. Além disso, suas categorias podem estabelecer a função de uma região de acordo com as atividades dos habitantes da mesma.

Desta forma, as regiões funcionais geradas pelo método proposto estarão sempre de acordo com a atualidade, facilitando estudos relacionados, que em sua maioria são custosos se realizados por meios tradicionais.

Tabela 1. Exemplos de locais existentes nas categorias do Foursquare.

Categoria do Foursquare	Exemplos de locais
Arts and Entertainment	Cinemas, museus e casinos.
College and University	Escolas, laboratórios universitários e centros de estudo.
Food	Restaurantes, cafeterias e padarias.
Nightlife Spot	Bares, clubes de *rock* e boates.
Outdoors and Recreation	Parques, academias e praias.
Professional and Other Places	Fábricas, auditórios e centros médicos.
Shop and Service	Lojas, salões de beleza e supermercados.
Travel and Transport	Aeroportos, estradas e hotéis.
Residence	Propriedades privadas, prédios e reboques residenciais.

A área de estudo foi definina na cidade de Nova Iorque e regiões adjacentes, gerando uma lista P com 2.208 polígonos com lados de 1 quilômetro de comprimento. Em relação aos períodos de tempo em T, foram definidos 6, sendo eles: 00:00 a 03:59, 04:00 a 07:59, 08:00 a 11:59, 12:00 a 15:59, 16:00 a 19:59 e 20:00 a 23:59.

Escolhemos polígonos de 1 quilômetro de comprimento para uma análise detalhada das regiões. Já as seis faixas de horários foram escolhidas para que fosse possível verificar como as regiões mudam sua função ao longo do dia de acordo com a rotina dos habitantes. Ressaltamos que tanto o tamanho das células quanto os horários são configuráveis.

4.1. Descrição dos Dados

Para a realização dos experimentos, foram coletados 157.054 *check-ins* do Foursquare entre abril de 2014 e agosto de 2014. Estes dados foram coletados no respectivo ano para posterior validação do método, que foi comparado com dados governamentais do mesmo período. Na Figura 2, apresentamos a quantidade de *check-ins* por categoria do Foursquare com os mesmos períodos de tempo contidos em T.

Podemos observar que a categoria com o maior número de *check-ins* é *Food*, enquanto que a menor é *Residence*. Isto ocorre porque muitos usuários evitam registrar suas propriedades privadas em redes sociais por motivos de segurança. Apesar disso, podemos observar nos resultados que algumas regiões funcionais são referentes à áreas residenciais.

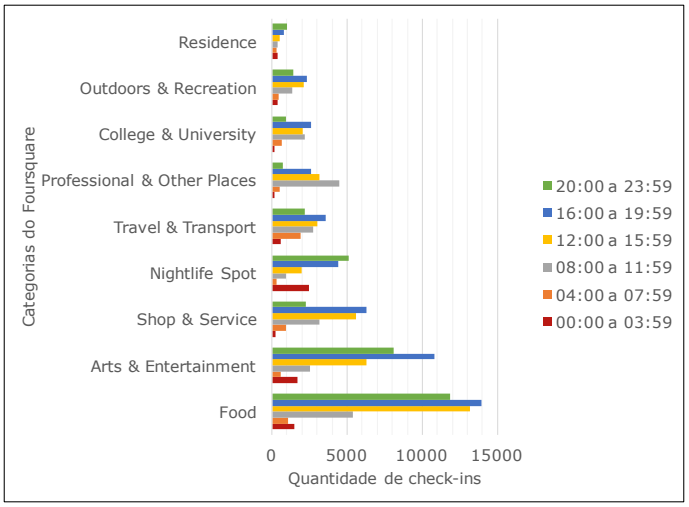


Figura 2. Quantidade de *check-ins* por categoria do Foursquare ao longo do dia.

4.2. Comparação com Dados Governamentais

Após a utilização dos dados sociais, o método proposto gerou regiões funcionais que mudam suas funções ao longo do tempo, de acordo com as atividades dos habitantes das mesmas. Como exemplo, podemos verificar na Figura 3, as regiões funcionais estabelecidas nos períodos 00:00 a 03:59 e 12:00 a 15:59.

No primeiro período (Figura 3(a)), as categorias com mais *check-ins* foram *Nightlife Spot*, *Arts & Entertainment* e *Food*, nesta ordem. Isto é refletido nas regiões funcionais estabelecidas. Da mesma forma, no segundo período (Figura 3(b)), as categorias com mais *check-ins* foram *Food* e *Shop & Service*, nesta ordem.

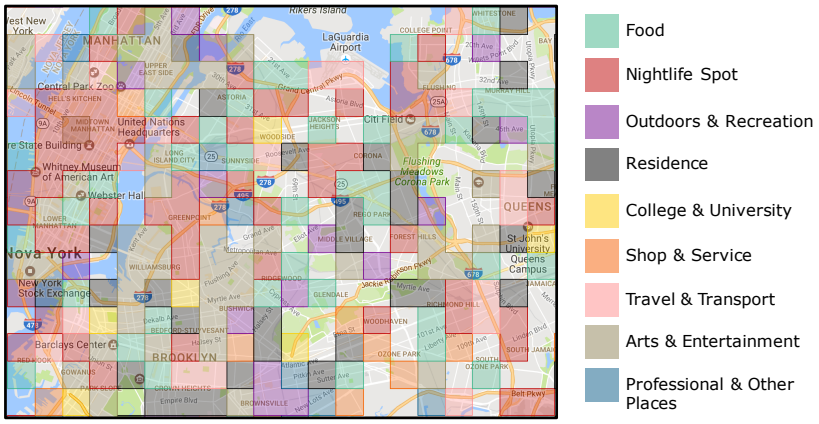
Além disso, podemos verificar que algumas áreas estão vazias. Isto ocorre porque, nestes períodos, não houveram dados sociais que refletissem a função da região. Isto pode ser resolvido aumentando-se a quantidade de dados sociais para estudo. Também é possível verificar o resultado da interpolação das regiões, que apresenta os polígonos de P' em diferentes formas.

Através destes resultados, podemos verificar que o método proposto permite que possamos realizar diferentes estudos de uma região, pois a função da mesma se altera de acordo com a época do ano e o período do dia, de forma que reflete a rotina dos usuários que nela residem.

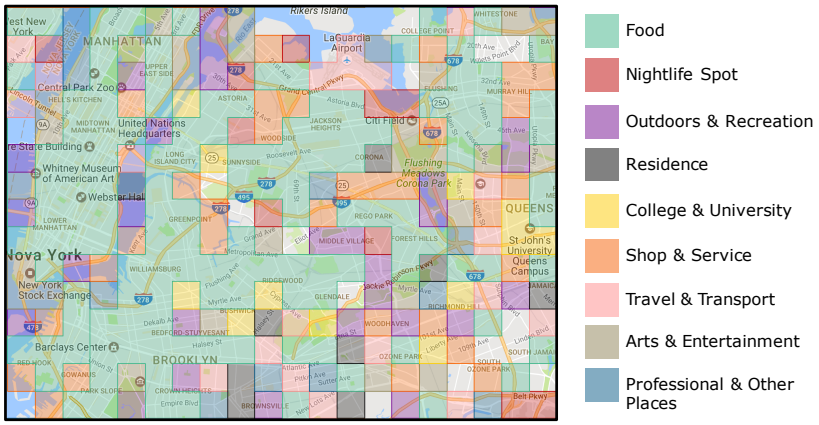
Para a validação do método proposto, foram utilizados dados do Departamento de Planejamento Urbano da cidade de Nova Iorque, denominados *City Owned and Leased Properties*[3]. Estes dados podem ser visualizados na aplicação *Zoning and Land Use Application*[4].

Os dados foram fornecidos em forma de uma planilha e estavam divididos por categorias que podem ser mapeadas para as categorias do Foursquare. Porém, os dados não eram geolocalizados, tornando necessária a utilização de uma API de mapa para transformar os endereços dos locais em coordenadas.

[3]`http://www1.nyc.gov/site/planning/about/publications/colp.page`
[4]`http://maps.nyc.gov/doitt/nycitymap/template?applicationName=ZOLA`



(a) 00:00 a 03:59



(b) 12:00 a 15:59

Figura 3. Resultado do método proposto para diferentes períodos de tempo, apresentando a mudança nas funções das regiões.

Em seguida, estes dados governamentais foram inseridos nas regiões funcionais cujas coordenadas geográficas coincidiam. Por fim, mapeamos as categorias de locais nos dados governamentais para as categorias macro do Foursquare, de acordo com a hierarquia de categorias desta rede social[5]. A partir disso, criamos e comparamos as regiões funcionais de ambas as fontes de dados. Regiões idênticas eram contadas como acerto, enquanto regiões diferentes eram contadas como erro. Desta forma, apuramos que o nosso método possui uma acurácia de 86% para a caracterização de uma área urbana.

Com a similaridade encontrada, temos a contribuição de que o método proposto pode ser útil para entender o comportamento de cidades mesmo quando não se tem recursos necessários para realização de censos. Além disso, verificamos que os dados sociais podem ser utilizados como meio alternativo para o auxílio no planejamento urbano de uma cidade.

5. Utilização das Regiões Funcionais

Nesta seção, apresentamos duas possíveis aplicações para o método proposto, a análise de acidentes de trânsito e o monitoramento dinâmico das cidades. Estas aplicações mos-

[5]https://developer.foursquare.com/categorytree

tram como a utilização de regiões funcionais pode facilitar a tomada de decisões e ações preventivas em uma determinada região, de forma dinâmica e com baixo custo. Apesar do foco nestas duas aplicações, o método proposto também pode ser empregado para a avaliação econômica, social e de mobilidade da área urbana estudada.

5.1. Aplicação I: Análise de Acidentes de Trânsito

Com o objetivo de criar soluções eficientes para a prevenção de acidentes de trânsito, órgãos governamentais buscam analisar os fatores relacionados aos mesmos em uma determinada região. Aspectos como a localização e o horário em que os acidentes ocorreram são levados em consideração, assim como as condições das vias, condutores e veículos.

Desta forma, uma possível aplicação para as regiões funcionais, além do auxílio ao planejamento urbano, é a verificação dos fatores relacionados aos acidentes de trânsito em cada região detectada que subdivide uma área urbana. Para isto, definimos as regiões funcionais através dos dados do Foursquare (conforme apresentado na seção 4). Em seguida, utilizamos dados de acidentes de trânsito fornecidos pelo departamento de polícia da cidade de Nova Iorque[6], com o objetivo de verificarmos a relação entre as regiões funcionais e os aspectos dos acidentes de trânsito.

Nestes dados do departamento de polícia, obtivemos informações sobre a data e a hora em que o acidente ocorreu, além da latitude, longitude, envolvidos no acidente e fatores que contribuíram para o mesmo. Após uma análise, divimos os fatores em 4 categorias:

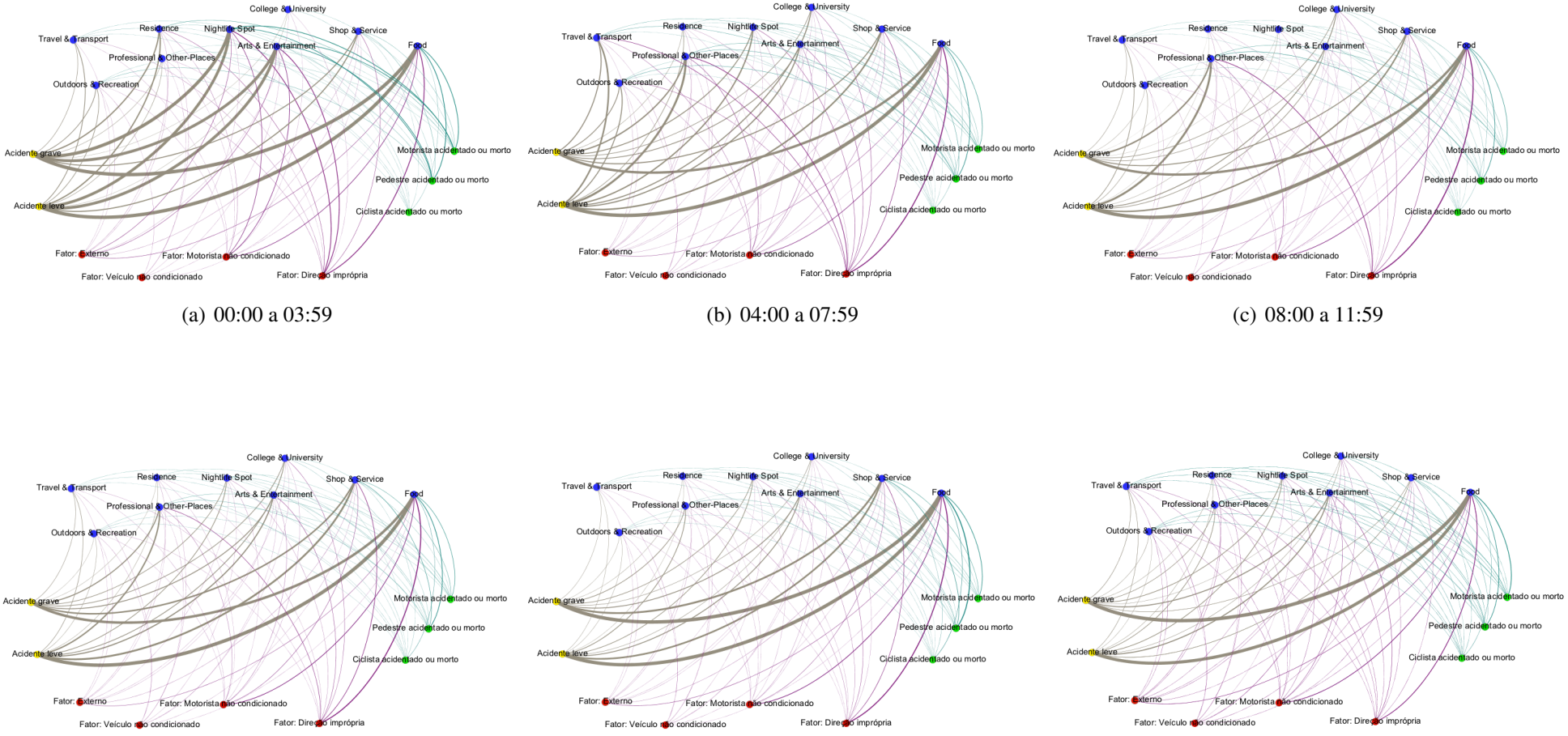
- **Direção imprópria**: ações como dirigir enquanto utiliza algum dispositivo móvel, fazer manobras proibidas na via e alta velocidade estão inclusas nesta categoria;
- **Motorista não condicionado**: condutores que estavam sob o efeito de bebidas alcoólicas ou drogas, estavam cansados ou perderam a consciência no momento do acidente estão inseridos nesta categoria;
- **Externo**: vias em condições ruins, animais e distrações provocadas por passageiros estão nesta categoria;
- **Veículo não condicionado**: falhas no acelerador, freios ou outras partes do veículo estão inseridas nesta categoria.

Além dos fatores descritos acima, também verificamos se o acidente ocasionou apenas danos materiais ou envolveu a morte do condutor ou de terceiros, como pedestres e ciclistas. No primeiro caso, o acidente foi classificado com **leve**. No segundo, como **grave**.

Como as regiões funcionais são representadas por polígonos com dados sociais relacionados a eles, aplicamos um algoritmo para verificar em qual região funcional os dados geolocalizados dos acidentes de trânsito (latitude e longitude) estavam inseridos. Assim, tornou-se possível associar os dados do departamento de polícia com os dados do Foursquare.

Desta forma, analisamos 73.182 acidentes de trânsito entre abril de 2014 e agosto de 2014, o mesmo período dos dados do Foursquare apresentados na seção 4. O resultado da análise foi dividido em 6 períodos de tempo para a verificação da mudança nas associações dos dados ao longo do dia (Figura 4).

[6] `https://data.cityofnewyork.us/Public-Safety/NYPD-Motor-Vehicle-Collisions/h9gi-nx95`



(a) 00:00 a 03:59 (b) 04:00 a 07:59 (c) 08:00 a 11:59

(d) 12:00 a 15:59 (e) 16:00 a 19:59 (f) 20:00 a 23:59

Figura 4. Representação da associação entre as regiões funcionais e os aspectos relacionados a acidentes de trânsito ao longo do dia.

Através das associações representadas na Figura 4 (sendo que quanto maior a incidência da associação, maior é a espessura da aresta), fizemos as seguintes observações relacionadas aos acidentes de trânsito:

- No período de 00:00 a 03:59 (Figura 4(a)), percebemos que a maioria dos acidentes, tanto graves quanto leves, estão associados às categorias *Nightlife Spot*, *Arts & Enterteinment* e *Food*, assim como os fatores *Motorista não condicionado* e *Direção imprópria*. Isto sugere que os acidentes nestas regiões estão relacionados ao uso de substâncias não permitidas, como o álcool, por exemplo. Além disso, verificamos que estes mesmos acidentes ocasionaram a morte de motoristas e pedestres.
- No período de 04:00 a 07:59 (Figura 4(b)), percebemos que os fatores relacionados aos acidentes começam a ser direcionados para outras categorias do Foursquare, como *Professional & Other Places*, *College & University* e *Shop & Service*. Isto está diretamente relacionado à rotina das pessoas na região estudada, que passam a realizar suas atividades cotidianas neste período de tempo.
- No período de 08:00 a 11:59 (Figura 4(c)), percebemos que os acidentes passam a se concentrar nas categorias *Professional & Other Places* e *Food*, devido ao horário do *rush*, e que a maioria dos acidentes está relacionado a direção imprópria. Neste período, o índice de acidentes leves é maior do que o de acidentes graves, o que indica que, na maioria dos acidentes, ocorrem apenas danos materiais. Um padrão semelhante é apresentado na Figura 4(d).
- No período de 16:00 a 19:59 (Figura 4(e)), percebemos que a maioria dos acidentes estão concentrados nas categorias *Food* e *Shop & Service*, e estão relacionados aos fatores *Motorista não condicionado* e *Direção imprópria*. Além disso, o número de mortes de motoristas e pedestres é maior em relação aos períodos anteriores. Um padrão semelhante é apresentado na Figura 4(f), porém apenas com a categoria *Food*.

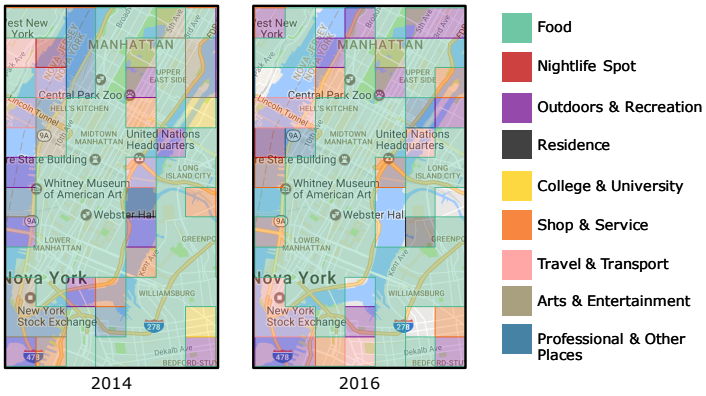
Esta aplicação é capaz de identificar os aspectos relacionados aos acidentes de trânsito e o que eles ocasionam, e pode ser utilizada para auxiliar na definição de soluções que minimizem os mesmos.

5.2. Aplicação II: Monitoramento dinâmico da cidade

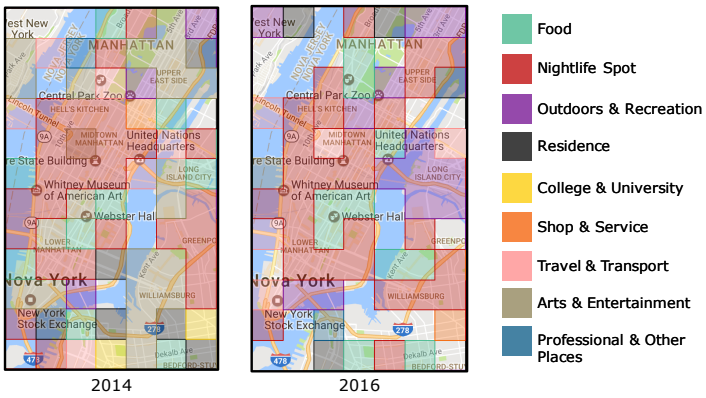
Com a detecção de regiões funcionais por meio de redes sociais, tornou-se possível que o levantamento de informações necessárias para o planejamento urbano fosse realizado dinamicamente, conforme a evolução das cidades. Além disso, este método facilita a execução desta tarefa, uma vez que não há dados e censos bem estabelecidos e com baixo custo.

Outro fator importante é que os levantamentos necessários são realizados entre longos intervalos de tempo, de 2 a 4 anos, por exemplo. Com os dados provenientes de redes sociais, as regiões funcionais podem ser definidas e alteradas conforme os usuários que habitam a região compartilham suas informações.

Para mostrar como este monitoramento dinâmico é viável, apresentamos a Figura 5, na qual é feita uma comparação da formação das regiões funcionais entre os anos de 2014 e 2016. Para isto, foram coletados 70.339 *check-ins* do Foursquare entre julho de 2016 e agosto de 2016 na mesma área urbana.



(a) 00:00 a 03:59



(b) 12:00 a 15:59

Figura 5. Comparação entre as regiões funcionais dos anos de 2014 e 2016 em diferentes períodos de tempo.

Podemos perceber, na Figura 5(a), que a categoria *Food* prevalece sobre as demais tanto no ano de 2014, quanto no ano de 2016. Isto é ocasionado devido ao horário em que as regiões funcionais foram formadas, de 12:00 a 15:59. Além disso, a presença de regiões funcionais com a categoria *Outdoors & Recreation* aumentou no ano de 2016 em relação ao ano de 2014. Também verificamos que, nas duas regiões funcionais em que anteriormente prevalecia a categoria *College & University*, agora prevalecem as categorias *Shop & Service* e *Outdoors & Recreation*. Estas mudanças podem ocorrer devido a expansão da cidade, construção de novos conjuntos residenciais e a criação de novos empreendimentos, por exemplo.

Na Figura 5(b), a comparação é realizada em um horário noturno, de 00:00 a 03:59. Neste horário, também verificamos o aumento na aparição de regiões funcionais com a categoria *Outdoors & Recreation*, apesar da categoria *Nightlife Spot* prevalecer sobre as demais. A categoria *College & University* também foi eliminada.

Desta forma, com esta aplicação, verificamos uma forma alternativa, de baixo custo, para obter as informações necessárias para o auxílio ao planejamento de uma área urbana. Isto torna possível não só que este planejamento seja realizado de forma dinâmica, como viabiliza que áreas urbanas com poucos recursos também consigam estas informações.

6. Conclusão e Trabalhos Futuros

Neste trabalho, apresentamos um método para a detecção de regiões funcionais através de redes sociais baseadas em localização. O método consiste na utilização de polígonos para a formação das regiões, além dos dados sociais e de intervalos de tempo. Desta forma, as regiões funcionais estão diretamente relacionadas a rotina dos habitantes da área urbana estudada, capturando inclusive sua dinâmica.

Como experimento, o método foi aplicado à cidade de Nova Iorque, e os resultados obtidos foram comparados com dados governamentais oficiais disponíveis. Com isso, apuramos que o método proposto obteve uma acurácia de 86% na definição das regiões funcionais, o que o torna uma alternativa viável para o auxílio no planejamento urbano em outras cidades onde não há tal monitoramento de forma sistemática. Suas principais vantagens são o pequeno custo de monitoramento e a possibilidade de observar o comportamento dinâmico das cidades e seus habitantes em intervalos de tempos muito menores que os métodos tradicionais. Não apresentamos experimentos com outras localidades, como as cidades brasileiras, porque apesar de ser possível obter dados do Foursquare para as regiões funcionais, não haveria como validar os resultados obtidos, pois os dados governamentais disponíveis são poucos, além de serem dispersos e incompletos.

Por fim, apresentamos duas aplicações para a qual as regiões funcionais podem ser utilizadas, além do planejamento urbano. Em uma das aplicações, é apresentado um estudo sobre os aspectos de acidentes de trânsito, relacionando-os com as regiões funcionais encontradas. Na outra, é apresentado como o monitoramento dinâmico de uma área urbana pode ser realizado através do método proposto.

Como trabalhos futuros, pretendemos analisar outras formas de definir a função predominante de uma região e verificar outras redes sociais baseadas em localização, que podem complementar os resultados encontrados, tendo em vista que diferentes países e cidades podem utilizar mais uma rede social do que outra. Além disso, iremos demonstrar através de outras aplicações a extensão deste estudo para avaliações econômicas, sociais e de mobilidade urbana.

7. Referências

Aggarwal, C. and Abdelzaher, T. (2013). Social sensing. *Managing and Mining Sensor Data*, pages 237–297.

Aiello, L., Petkos, G., Martin, C., Corney, D., Papadopoulos, S., Skraba, R., Göker, A., Kompatsiaris, I., and Jaimes, A. (2013). Sensing trending topics in twitter. *IEEE Transactions on Multimedia*, 15:1268–1282.

Almeida, T., Souza, B., Menezes, A., Figueiredo, C., and Nakamura, E. (2016). Sentiment analysis of portuguese comments from foursquare. In *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web*.

Cranshaw, J., Schwartz, R., Hong, J., and Sadeh, N. (2012). The livehoods project: Utilizing social media to understand the dynamics of a city. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*.

Liu, Y., Liu, X., Gao, S., Gong, L., Kang, C., Zhi, Y., Chi, G., and Shi, L. (2015). Social sensing: A new approach to understanding our socioeconomic environments. *Annals of the Association of American Geographers*, 105:512–530.

Madan, A., Cebrian, M., Lazer, D., and Pentland, A. (2010). Social sensing for epidemiological behavior change. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*.

Menezes, A., Almeida, T., Gatto, B., Santos, E., Figueiredo, C., and Nakamura, E. (2016). Poi: uma aplicação de detecção de pontos de interesse. In *Proceedings of the 13th Simpósio Brasileiro de Sistemas Colaborativos (SBSC)*.

Moturu, S., Khayal, I., Aharony, N., Pan, W., and Pentland, A. (2011). Using social sensing to understand the links between sleep, mood, and sociability. In *Proceedings of 3th IEE International Conference on Social Computing Privacy, Security, Risk and Trust (PASSAT)*.

Noulas, A., Scellato, S., Mascolo, C., and Pontil, M. (2011a). An empirical study of geographic user activity patterns in foursquare. In *Proceedings of the 15th International AAAI Conference on Weblogs and Social Media*.

Noulas, A., Scellato, S., Mascolo, C., and Pontil, M. (2011b). Exploiting semantic annotations for clustering geographic areas and users in location-based social networks. In *Proceedings of the 15th Internation AAAI Conference on Weblogs and Social Media*.

Sakaki, T., Okazaki, M., and Matsuo, Y. (2010). Earthquake shakes twitter users: Real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web*.

Shi, J. and Malik, J. (1997). Normalized cuts and image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.

Silva, T., Vaz de Melo, P., Almeida, J., Salles, J., and Loureiro, A. (2013). A comparison of foursquare and instagram to the study of city dynamics and urban social behavior. In *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing*.

Silva, T. H., de Melo, P. O. V., Almeida, J. M., Salles, J., and Loureiro, A. A. (2012). Visualizing the invisible image of cities. In *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, pages 382–389. IEEE.

Steenbruggen, J., Tranos, E., and Nijkamp, P. (2015). Data from mobile phone operators: A tool for smarter cities? *Telecommunications Policy*, 39:335–346.

Traynor, D. and Curran, K. (2012). Location-based social networks. *From Government to E-Governance: Public Administration in the Digital Age*, page 243.

Xiang, F., Tu, L., Huang, B., and Yin, X. (2013). Region partition using user mobility patterns based on topic model. In *Proceedings of the 16th IEEE International Conference on Computational Science and Engineering (CSE)*.

Zhang, K., Jin, Q., Pelechrinis, K., and Lappas, T. (2013). On the importance of temporal dynamics in modeling urban activity. In *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing*.

Zhi, Y., Li, H., Wang, D., Deng, M., Wang, S., Gao, J., Duan, Z., and Liu, Y. (2016). Latent spatio-temporal activity structures: a new approach to inferring intra-urban functional regions via social media check-in data. *Geo-spatial Information Science*, pages 1–12.

T-MAPS: Modelo de Descrição do Cenário de Trânsito Baseado no Twitter

Bruno P. Santos[1], Paulo H. L. Rettore[1], Heitor S. Ramos[2], Luiz F. M. Vieira[1], Antonio A. F. Loureiro[1]

[1] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte – MG – Brasil

[2]LACCAN/LCCV/Instituto de Computação
Universidade Federal de Alagoas (UFAL) – Maceió – AL – Brasil

[1]{bruno.ps, rettore, lfvieira, loureiro}@dcc.ufmg.br

[2]heitor@ic.ufal.br

Abstract. *Understanding urban mobility, specifically transit mobility, has been focous of many research and investiments. However, obtain free access to real traffic data it is still limited, because the data is usually private and install sensors on transport network is expensive. In this paper, we caracterize the use of a Location Based Social Media (LBSM) plataforms to obtain a cost efficient transit network sketch. Also, we propose Twitter MAPS (T-MAPS) a description model of the transit scenario based on tweets. T-MAPS aim to improve todays' navegation systems and urban planning with information from LBSMs. In the avaluation, T-MAPS suggested routes with, in average, 62% of similarity with Google Directions' routes. Also, in 25% of the routes avaluated, the similarity reached 87% to 100%. These results indicates significant relationship between T-MAPS and Google Directions routes, even T-MAPS using solely tweets.*

Resumo. *A compreensão da mobilidade urbana, em específico do trânsito, tem sido alvo de estudos e investimentos. Contudo, a obtenção de dados que representam o cenário do trânsito apresenta limitações, seja de acesso aos dados ou pelo elevado custo em implantar sensores na malha viária. Neste artigo, foi caracterizado o uso uma plataforma LBSM (Twitter), a fim de se obter uma descrição, de baixo custo, do trânsito. Também foi proposto o Twitter MAPS (T-MAPS), um modelo de descrição do cenário de trânsito baseado em tweets, com o objetivo de enriquecer as navegações e planejamento urbano com informações sobre regiões. Na avaliação, as sugestões de rotas do T-MAPS mostraram-se, em média, 62% similares às do Google Directions e, em 25% dos casos, a similaridade observada foi entre 87% e 100%, o que indica significativa relação entre as abordagens, mesmo T-MAPS usando somente tweets.*

1. Introdução

A economia e qualidade de vida em uma cidade é, em parte, reflexo da mobilidade que ela oferece. A infraestrutura de transporte local deve possibilitar o deslocamento das pessoas, o recebimento de insumos e escoamento das produções das empresas de forma eficiente.

Isso implica na necessidade de constante planejamento, gerência e manutenção dos sistemas de transporte. Nesse aspecto, a compreensão da mobilidade urbana, em específico da mobilidade no trânsito, tem despertado o interesse dos órgãos governamentais de planejamento e da academia. Contudo, para compreender o cenário de trânsito, é fundamental a obtenção e acesso a dados, tais como contagem de veículos passando por *loops* indutivos ou câmeras de trânsito, *traces* de usuários nas vias de transporte, matrizes de origem e destino de determinada região, dentre outros. O acesso a estes dados representa desafios na compreensão do cenário de trânsito de uma região, pois grande parte dos dados são controlados por entidades privadas ou governamentais e, geralmente, são inacessíveis em parte ou em sua totalidade, seja por questões de mercado ou mesmo privacidade. Como alternativa de baixo custo ao cenário atual para obtenção de dados, surge a oportunidade de usar as mídias sociais baseadas em localização – *Location Based Social Media (LBSM)*) como, por exemplo, *Twitter* e *Foursquare*, com o objetivo de suprir a deficiência de dados sobre um contexto específico.

As LBSMs são amplamente utilizadas por usuários que fornecem os dados, sejam eles corporativos ou não. Segundo o *Twitter*, cerca de 313 milhões de usuários estão ativos todos os meses (dados de junho de 2016) em sua rede. Isto pode gerar diversas oportunidades em busca de melhor compreender uma determinada situação do trânsito, por exemplo. Contudo, os dados geralmente apresentam particularidades, inerentes ao uso pelas pessoas, que podem acarretar em grandes desafios como, por exemplo, dados imprecisos, enviesados e inconsistentes.

Neste trabalho, um estudo piloto foi realizado para melhor entender o relacionamento entre o cenário real do trânsito e os dados de uma plataforma LBSMs, o Twitter. A partir desse estudo, os detalhes das características dos dados foram levantadas e classificadas. Após essa caracterização, foi desenvolvido o Twitter MAPS (T-MAPS), um modelo de descrição do cenário de trânsito baseado no Twitter, com o objetivo de enriquecer o contexto de navegação atual, vinculando os *tweets* às rotas geradas por plataformas já consolidadas como, por exemplo, o *Google Directions*. O modelo é apresentado de forma generalista e pode ser aplicado a qualquer região. No entanto, neste trabalho os dados e, consequentemente, os resultados são provenientes do estado de Nova Iorque nos Estados Unidos.

As principais contribuições realizadas neste trabalho são:

- Caracterização de redes LBSM, em especial do Twitter, como uma fonte de dados para melhor entender e descrever o cenário de trânsito.
- Proposição do T-MAPS como um modelo de descrição cenário de trânsito baseado em dados extraídos do Twitter.

A motivação deste trabalho vem do desejo em ampliar o conhecimento do trânsito em uma dada região, usando dados obtidos de LBSMs, com o objetivo de proporcionar uma experiência mais descritiva desse cenário, pouco explorado ainda na literatura. A indicação do sentimento de um trajeto, a intensidade de acidentes e descrições mais detalhadas de ocorrências são exemplos de informações que podem enriquecer a experiência do usuário da infraestrutura de transporte e melhor auxiliá-lo nas decisões relacionadas a mobilidade urbana.

O restante deste trabalho está organizado como descrito a seguir. A Seção 2 apresenta as características da coleta de dados do *Twitter*. A Seção 3 descreve os problemas

Tabela 1. Exemplo de contas

Nome da Conta	# *Tweets*
@511NYC	126925
@TotalTrafficNYC	20267
@WazeTrafficNYC	7850
@Traffic4NY	3789
...	...
@NYC_DOT	3680
Total das 21 contas:	355K

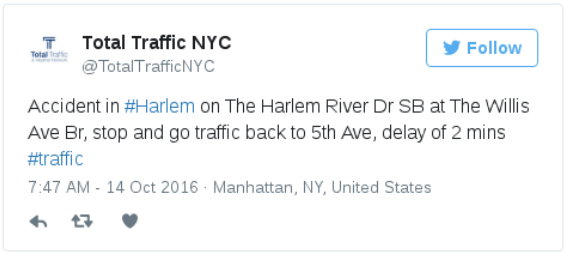
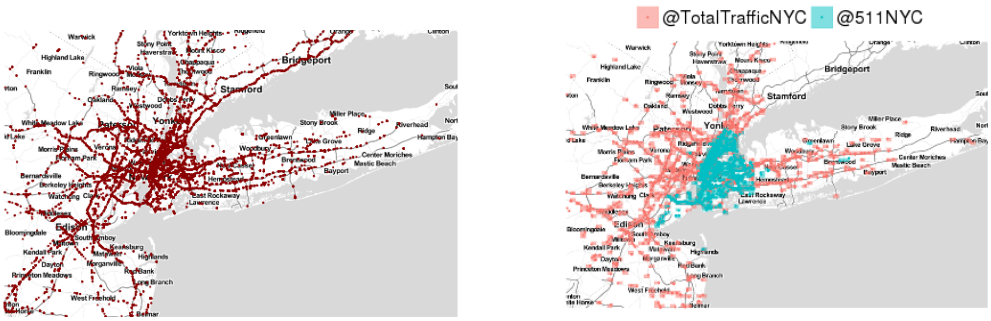


Figura 2. Exemplo de *Tweet*

existentes nesses dados, os quais devem ser tratados para serem utilizados. A Seção 4 descreve os passos da modelagem T-MAPS, que é o resultado da utilização desses dados. A avaliação da modelagem é realizada na Seção 5. Em seguida a Seção 6 apresenta os trabalhos relacionados e, por fim, a Seção 7 apresenta a conclusão e trabalhos futuros.

2. Coleta de Dados

O processo de coleta dos dados do Twitter foi conduzido utilizando a REST Aplication Program Interface (API)[1], seguindo os termos de limites da plataforma. Foram selecionados, manualmente, 21 contas especializadas em notificar o trânsito em todo o estado de Nova Iorque por meio de *tweets*. Essas contas são dedicadas apenas à publicação de eventos de trânsito através de *tweets* e são mantidas por diferentes organizações, incluindo órgãos governamentais. No período de tempo amostrado, às 21 contas reportaram um montante de aproximadamente 355 mil *tweets*. A quantidade de dados que apresentam etiquetas de geolocalização ultrapassa 307 mil e os dados que estão posicionados dentro da região de Manhattan é de aproximadamente 38 mil. Um exemplo de *tweet* publicado por uma das contas especialistas em trânsito é apresentado na Figura 2. É possível notar a variedade de informações disponibilizadas neste *tweet*; tem-se desde a descrição do evento ocorrido (um acidente) até horário, informações sobre congestionamento, direção e localização precisa. É importante notar que as contas de usuários comuns não foram consideradas devido ao maior viés de seus *tweets* em relação ao trânsito.



(a) *Tweets* na cidade de Nova Iorque (b) Cobertura espacial de duas contas

Figura 3. Cobertura de espacial

[1] https://dev.twitter.com/rest/public

A Figura 3 apresenta a cobertura espacial dos dados coletados na cidade de Nova Iorque e regiões adjacentes e, em específico, a Figura 3(a) destaca a cobertura de *tweets* com geolocalização coletados de todas as 21 contas especialista. Ao aproximar a imagem é possível notar que os *tweets* estão associados às vias na malha de transporte da cidade. A Figura 3(b) mostra a cobertura espacial de duas contas especialistas, @TotalTrafficNYC e @511NYC, as quais possuem maior número de publicações. O maior raio de cobertura é apresentado pela conta @TotalTrafficNYC, publicando *tweets* tanto no perímetro da cidade de Nova Iorque quanto nas regiões adjacentes, ao passo que a conta @511NYC limita-se à cidade. Esses dados apresentam a informação de que diferentes contas possuem diferentes áreas de cobertura. Cientes disto, é possível selecionar contas que se complementam aumentando, assim, a cobertura espacial de uma região de interesse.

Entender a cobertura temporal é outro ponto importante ao manipular dados de LBSMs. A Figura 4 mostra como três contas especialistas se comportam ao longo de uma semana. A conta @NYC_DOT está associada ao departamento de transporte da cidade de Nova Iorque e apresenta um típico comportamento de trabalho em horário comercial, não publicando *tweets* durante os finais de semana. Já @TotalTrafficNYC e @511NYC apresentam comportamento constante durante as semanas, mas variam as taxas de atividade ao longo de um dia, como mostrado na Figura 4 mais à direita. Essas informações também podem ser uteis para combinar contas e aumentar a cobertura temporal.

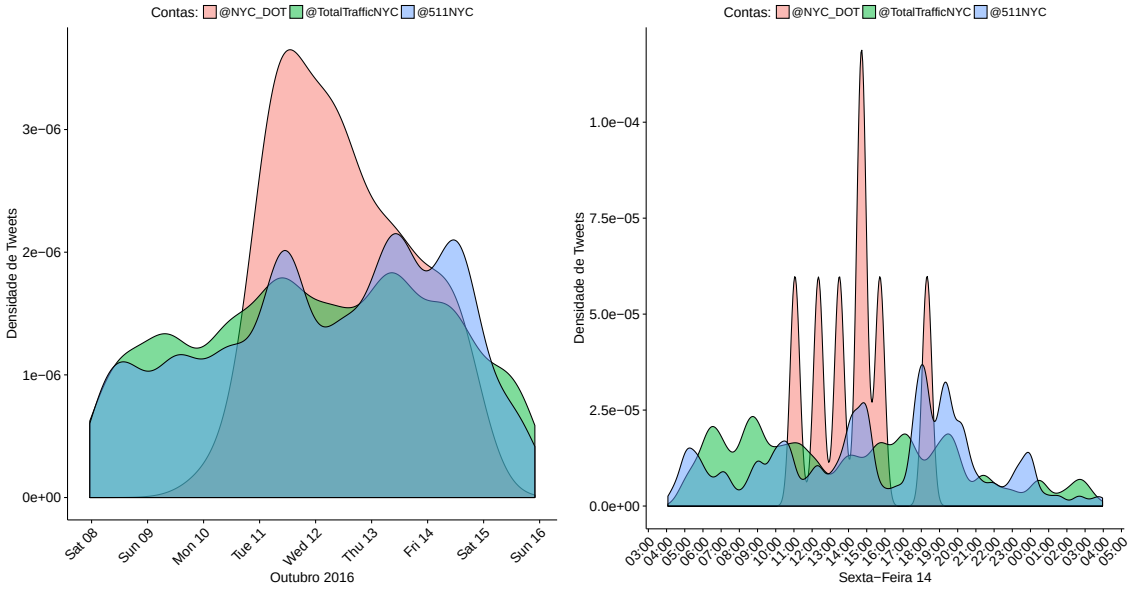


Figura 4. Cobertura temporal de três contas especialistas

Para complementar os pontos levantados anteriormente, a Figura 5 apresenta uma visão espacial e temporal dos dados coletados na Sexta-Feira, 14 de Outubro de 2016. Os dados foram agregados em faixas de uma hora, sendo possível observar que nas primeiras horas ocorreram poucos *tweets* na cidade de Nova Iorque. Por outro lado, é perceptível o aumento das atividades após as 6 horas da manhã, momento em que a população tende a iniciar suas atividades. Após às 12 horas, também é possível notar o aumento na quantidade de atividade, em especial, nos horários de 15 e 18 horas e, logo após esse horário, existe uma tendência de diminuição do número de *tweets* reportados por hora, até voltar

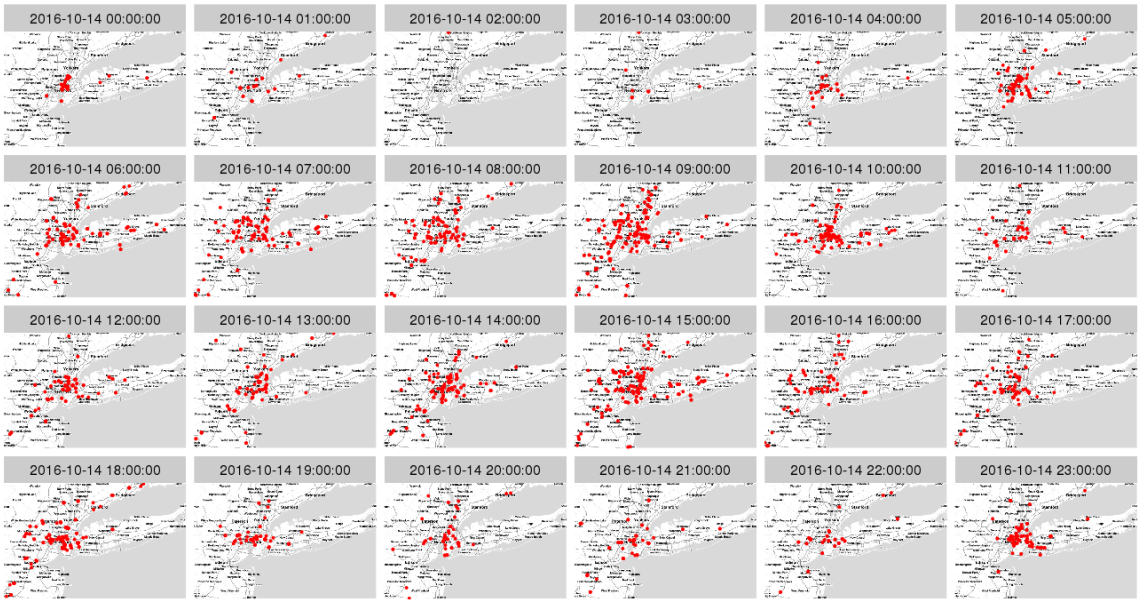


Figura 5. Cobertura espaçotemporal

a aumentar às 23 horas. Isso se deve ao fato do dia da semana analisado, pois supõe-se que esse aumento das atividades, especialmente concentrada em Manhattan, esteja relacionada ao início de atividades noturnas, típicas do início do final de semana.

2.1. Correlação entre *tweets* e trânsito

Identificar o quão bem relacionados estão os dados de LBSMs com o trânsito pode revelar a capacidade que tais dados têm de representar o cenário do trânsito. No estudo aqui apresentado, essa informação de relacionamento pode dizer se os *tweets* são relevantes para descrever, de algum modo, o cenário do trânsito no tempo. Por esse motivo, esta seção visa responder à seguinte pergunta: *os dados coletados do Twitter apresentam correlação com dados reais de trânsito?*

Para responder a esta pergunta é necessário obter dados reais do trânsito, tais como contagem de veículos passando por *loops* indutivos ou câmeras de trânsito, rastro de GPS de usuários na rede de transporte, matrizes de origem e destino de uma região, dentre outras. Esses dados serão considerados como dados tradicionais, os quais permitem estudar as interações entre demanda (motoristas, veículos, pedestres, ciclistas) e a infraestrutura (ruas, rodovias, sinais, dispositivos de controle, etc.). Essas interações ajudam a entender e desenvolver malhas de transporte mais eficientes, que otimizam o movimento do tráfego e reduzem congestionamentos [Bazzan and Klügl 2013].

Os dados tradicionais de trânsito têm por finalidade capturar três principais variáveis: velocidade, densidade e fluxo. Essas variáveis permitem visualizar o comportamento do trânsito [Kung et al. 2014, Zheng et al. 2008, Bazzan and Klügl 2013]. No entanto, os dados tradicionais isoladamente não capturam a experiência dos usuários que utilizam a malha de transporte; o sentimento que se tem a respeito das regiões; ou eventos com maior teor semântico como a gravidade de um acidente. Além disso, acessar dados tradicionais pode não ser uma tarefa fácil, já que muitos deles são privados (corporativos ou governamentais) como, por exemplo, trajetos de GPS de *smartphones*, trajetos de GPS

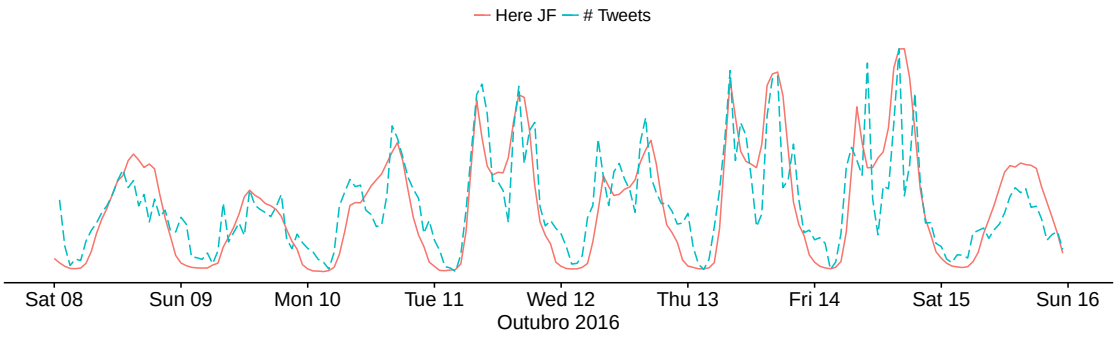


Figura 6. Número de ocorrências de *tweets* e o fator de congestionamento provido pela empresa *HERE*. A correlação de Spearman para as séries é de 0.81

veicular (ex: táxi) e *loops* indutivos.

Grandes empresas ou agências de transporte governamentais geralmente realizam processos de fusão sobre os dados observados. Esses processos têm por objetivo enriquecer o conhecimento sobre o estado atual do trânsito e facilitar sua visualização. Nessa direção algumas empresas como *HERE*, TomTom e *Google* permitem o acesso, *parcial*, aos seus mapas de trânsito que contêm o resultado do processo de fusão sobre os dados. Por exemplo, a empresa *HERE* fornece uma API de acesso ao dado *Jam Factor (JF)*[2] (fusão dos dados tradicionais e outros), o qual representa o nível de congestionamento das vias em que a empresa possui dados, ao passo que *Google* e *TomTom* não oferecem acesso direto a essa informação. Por esse motivo, o JF será utilizado no comparativo de relacionamento entre *tweets* e o contexto do trânsito.

A Figura 6 apresenta duas curvas em série temporal entre os dias 8 e 16 de Outubro de 2016. A curva tracejada representa a contagem de *tweets* ao longo do tempo, enquanto a curva sólida representa o fator de congestionamento fornecido pela API HERE (JF). As curvas foram normalizadas e estão em uma escala entre 0 e 1. É possível perceber o forte padrão periódico nessas séries de dados. Os dias da semana possuem maior contagem de *tweets* e fator de congestionamento quando comparado aos finais de semana. Esse padrão se repete em conjunto com dois pontos de maior contagem e JF.

O coeficiente de correlação de Spearman foi utilizado como medida de relacionamento entre os *tweets* e o fator de congestionamento. Essa correlação é mais adequada nesse cenário do que a tradicional correlação de Pearson, pelo fato das séries de dados não apresentarem comportamento linear. Tal como a correlação de Pearson, a de Spearman apresenta valores entre -1 e $+1$ para correlações perfeitas. O coeficiente resultante entre as duas séries de dados foi de *0.81*. A interpretação dessa correlação indica que os dados estão altamente relacionados e que, por ser uma correlação positiva, em momentos nos quais o fator de congestionamento cresce o número de *tweets* também tende a crescer.

3. Aspectos dos Dados

Observada a alta correlação entre os dados do *tweets* e o cenário do trânsito, esta seção apresenta alguns aspectos que os dados do Twitter e de LBSMs podem apresentar. Os dados geralmente apresentam particularidades que podem acarretar em dificuldades no

[2] https://developer.here.com/rest-apis/documentation/traffic/topics/quick-start.html

uso para determinar o cenário do trânsito. Os dados serão classificados em cinco aspectos e, em seguida, serão discutidos os problemas e potenciais soluções, sempre que possível.

3.1. Imprecisão

Dados providos pelos *tweets*, quando fazem referência a informações de trânsito, podem apresentar alguma imprecisão. Isso é revelado quando os dados apresentam ao menos uma das seguintes características: dados incompletos; vagos ou granularidade que afeta sua interpretação. É esperado que imprecisões apareçam nos dados causadas pela inerente heterogeneidade das fontes e do alto grau de liberdade da entrada de dados. A seguir, é apresentado um exemplo de *tweet* não geolocalizado que será utilizado para ilustrar os aspectos de imprecisão descritos:

"Agora 8:00AM um acidente na Av. Antônio Carlos #BH #trafégoRuim #asustado".

- **Incompleto:** é comum que *tweets* não apresentem informações completas do evento ocorrido, principalmente pela limitação de espaço do texto. No *tweet* do exemplo, é possível obter informações tais como sentimento do usuário, condição do tráfego (para aquele usuário) e o horário. Entretanto, os dados diretamente acessíveis são insuficientes (e.g., inexistência de etiqueta de geo-localização e descrição textual) para derivar facilmente a localização do evento. Uma possível forma de solução é a aplicação de técnicas de aprendizado de máquina ou *record-linkage*.
- **Vago:** está relacionado ao grau de clareza da informação sobre o evento e seu contexto. No *tweet* do exemplo, o dado está vago, pois ao extrair as informações do texto, pouco é dito sobre o local exato do incidente (a Av. Antônio Carlos, possui mais de 8 km de extensão). Neste caso, o *tweet* é vago em relação à localização do evento.
- **Granularidade:** varia de baixa a alta granularidade. Quando são de baixa granularidade, os dados contêm informações suficientes para descrever precisamente os seguintes itens: localização do evento, sentido, gravidade e outras informações de interesse. Caso contrário, são considerados de alta granularidade e apresentam uma visão macro dos eventos.

3.2. Viés dos usuários

Os dados obtidos e analisados pelas LBSMs podem conter vieses, uma vez que, as informações sobre trânsito e tráfego podem ser reportadas por qualquer usuário. Por exemplo, os congestionamentos podem ser percebidos de diferentes maneiras. Suponha o seguinte cenário onde o usuário de uma pequena cidade está no trânsito de uma grande metrópole: nesse caso, ele pode perceber o congestionamento de forma diferente de um usuário que reside nessa metrópole. Consequentemente, é introduzido ao dado a percepção do usuário, gerando problemas como inconsistência e conflitos.

Até mesmo usuários especialistas[3] podem introduzir vieses. Usuários dessa classe podem, por exemplo, manter um público alvo específico ou ainda apresentar maior ou menor intenção de publicar de informações de determinado local em detrimento de outros.

[3]Consideram-se como usuários especialistas as contas que têm como propósito reportar apenas informações sobre o cenário de trânsito.

No conjunto de dados explorados nesse trabalho, foram selecionados manualmente os usuários que representam contas corporativas, ou seja, entidades como jornais ou departamento de transporte local. Desse modo, apesar da natureza diversa dessas contas, os dados podem seguir vieses inerentes aos interesses dessas contas.

3.3. Atribuição espacial e temporal

A atribuição espacial e temporal são os pontos mais críticos dos aspectos dos dados providos por LBSMs. A geo-localização e o aspecto temporal dos dados permitem caracterizar uma dada região e um instante ou intervalo de tempo. Por outro lado, os dados que não apresentam esses aspectos podem não fazer sentido para a construção do cenário atual do trânsito. A seguir, serão descritos os principais detalhes da atribuição espacial e temporal no processo de extração dessas informações de uma LBSMs:

- **Espacial:** atribuir uma localização ao dado é fundamental para entender o contexto que envolve a informação. Contudo, derivar essas informações, mesmo estando presente nos dados ou meta-dados dos *tweets*, não é uma tarefa trivial. Suponha, por exemplo, que a informação da posição no espaço esteja incorporada ao texto de um *tweet*. Portanto, extrair tal informação requer uma estrutura que possibilite a identificação desses dados no texto. Entretanto, o texto de um *tweet* é inerentemente desestruturado, permite poucos caracteres e existem formas diferentes de se escrever o texto, o que resulta, muitas vezes, em subjetividade na interpretação das informações (e.g., "Avenida" e "Av.", "R. Nome" e "R." significa Rua ou Rodovia?). Desse modo, técnicas de *Natural Language Processing (NLP)* [Liu et al. 2011, Li and Sun 2014] têm obtido resultados interessantes na extração de tais informações. Em [Li and Sun 2014], os autores apresentam uma técnica baseada em NLP, chamada *Petar*, para extrair Point of Interests (POIs) dos textos dos *tweets*.
  Disponibilidade de informação é outro ponto que afeta a atribuição espacial. Espera-se que algumas regiões tenham maior cobertura espacial que outras por diversos fatores. Por exemplo, grandes cidades tendem a apresentar maior cobertura espacial do que cidades pequenas.
- **Temporal:** associar um aspecto temporal (*timestamp*) ao dado publicado é de suma importância para entender o cenário passado, atual e, possivelmente, futuro da malha de transporte. Em geral, as plataformas LBSM associam uma marca de tempo no momento em que a informação é reportada. Entretanto, essa marcação não necessariamente é a mesma do momento em que o evento ocorreu. Assim, surgem alguns questionamentos em relação a esse aspecto tais como: *Qual é a validade de um dado publicado por um usuário de LBSM? Como se caracteriza o atraso entre o evento e o surgimento da informação nas plataformas LBSM?*

3.4. Inconsistências

Nesta seção, são abordadas questões que podem gerar inconsistências no uso de dados de plataformas LBSM. A noção de inconsistência diz respeito aos aspectos de conflito e desordem dos dados. Uma visão semelhante é apresentada em [Rettore et al. 2016].

- **Conflito:** os dados de LBSMs são conflitantes quando duas ou mais fontes de informação divergem a respeito de um evento. Por exemplo, suponha que dois usuários do Twitter publiquem *tweets* a respeito do mesmo evento, como

um possível incidente. Um dos usuários relata que nada de grave ocorreu e o fluxo de trânsito está bom, enquanto o outro usuário reporta um grave acidente que gerou impacto negativo no trânsito. Nesse caso, somente com essas duas informações não se pode dizer se, de fato, ocorreu um acidente e nem as consequências do evento. Soluções baseadas em teoria da evidência (Dempster-Shafer) têm ganhado notoriedade em reduzir divergências de dados conflitantes [Zadeh 1984, Florea et al. 2009]. Outra abordagem, mais simplista, é empregar diferentes pesos às informações providas por diferentes contas de usuários, por exemplo, usuários comuns recebem peso X enquanto usuários especialistas recebem peso Y e, assim, aplicar regras para decidir sobre as informações.

- **Fora de ordem:** a liberdade oferecida pelas plataformas LBSMs permitem aos usuários inserir dados fora de ordem, no tempo. Esses dados aparecem como inconsistentes para os sistemas que venham a utilizá-los. Esse aspecto se relaciona como o aspecto temporal dos dados, e a principal questão que surge é como empregá-los de forma adequada. Uma possível solução é descartar os dados que estão fora de ordem. Entretanto, tal solução implica em perda de informação e, consequentemente em degradação da cobertura espacial e temporal do sistema. Uma segunda alternativa é armazenar todos os dados obtidos e ordená-los, necessitando mais recursos computacionais como processamento e armazenamento.

4. Processo de modelagem do Twitter MAPS (T-MAPS)

Esta seção apresenta uma modelagem que permite utilizar os dados coletados (veja a Seção 2) do Twitter como uma representação do cenário de trânsito. A modelagem proposta foi desenvolvida de forma flexível, com o objetivo de possibilitar a agregação de dados de outras plataformas LBSMs. O processo de modelagem é descrito a seguir:

- ***Passo 1 – Aquisição de informações:*** essa etapa consiste em segmentar a área de interesse e obter dados das plataformas LBSMs. A segmentação é realizada devido as inerentes lacunas na cobertura espacial dos dados, o propósito desta segmentação é viabilizar uma visão macro do trânsito local a partir da associação dos dados com o mapa.
- ***Passo 2 – Filtragem e fusão de dados:*** cria-se um grafo ponderado e dirigido para representar a segmentação realizada. O grafo é definido como $G = (V, A)$, tal que, $V(G)$ é o conjunto de vértices que representam as divisões da área e $A(G)$ é o conjunto de arestas entre quaisquer dois vértices de G, desde que as áreas que eles representam sejam adjacentes no mapa real.
- ***Passo 3 – Métricas de custo:*** esse passo consiste em atribuir pesos às arestas direcionadas da seguinte forma: $A(x, y) \leftarrow \text{peso}(y)$. Esse peso representa a descrição do cenário do trânsito.

A Figura 7 ilustra as etapas da modelagem proposta e as diferentes formas de ponderar o grafo. Iniciando com o mapa de regiões de Nova Iorque, obtém-se o grafo $G = (V, A)$, sendo $V(G)$ as regiões e $A(G)$ as adjacências. Em seguida, a camada de dados do *Twitter* é aplicada ao grafo com o objetivo de posteriormente acrescentar informações de mídias sociais como peso às arestas do grafo. O modo utilizado para ponderar o grafo G consiste em $A(v, w) \leftarrow \text{peso}(w)$, onde $\text{peso}(w)$ pode seguir várias estratégias. Abaixo são listadas algumas dessas estratégias:

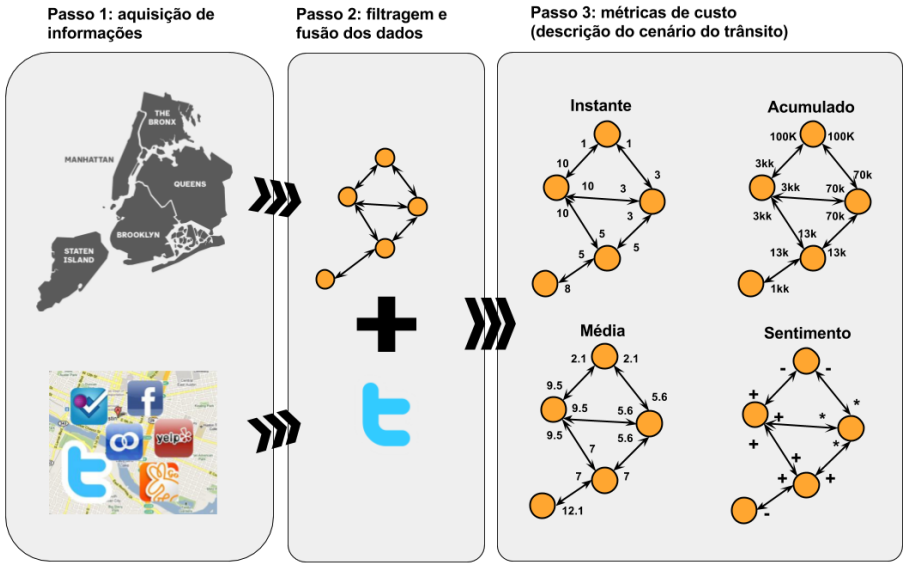


Figura 7. Processo de modelagem

- **Instante:** abordagem que considera os *tweets* de um dado dia e instante no tempo, correspondendo ao agregado de uma hora fechada. Esta estratégia pode ser considerada como uma caricatura do trânsito naquele momento.
- **Acumulado:** consiste na agregação dos *tweets* ao longo do tempo, desde que correspondam ao mesmo dia da semana e hora fechada. Nesse caso, leva-se em conta o histórico daquela região para ponderar as arestas do grafo.
- **Média:** apropria-se da mesma abordagem do *Acumulado*, contudo os *tweets* representam a média das ocorrências ao longo do tempo, dia da semana e hora fechada. Essa estratégia pode ser utilizada como cenário padrão médio do trânsito e sempre que novos dados entram no sistema, pode-se comparar com a média histórica e, então inferir sobre o trânsito.
- **Sentimento:** o texto do *tweet* é analisado, com o objetivo de associar um sentimento a esse texto. Com essa atribuição, pode-se considerar o sentimento (e.g., positivo, negativo ou neutro) que mais influencia cada região.

A modelagem desenvolvida, apresentada como T-MAPS, busca descrever o cenário de trânsito baseando-se nos *tweets* de uma dada região. Apoiado nesse processo, o T-MAPS busca prover serviços que auxiliem os usuários a compreender o trânsito de maneira mais abrangente. Dentre os serviços propostos pelo T-MAPS, destacam-se: prover uma visão macro do trânsito, indicar o sentimento das regiões ao longo do tempo, sugerir rotas baseadas na frequência de ocorrência registradas nos *tweets*, sugerir rotas baseadas no sentimento das regiões e destacar rotas com elevado número de acidentes ou desastres. É importante salientar que, ferramentas como *Google/Microsoft/TomTom* exploram outros aspectos na sugestão das rotas de trânsito. Assim, os serviços do T-MAPS constituem uma alternativa para enriquecer os cenários que se têm do trânsito com informações publicadas em plataformas LBSMs.

5. Avaliação do T-MAPS

Nesta seção, são apresentadas avaliações do T-MAPS. Inicialmente o serviço de rotas do T-MAPS é avaliado utilizando o *Google Directions*, aqui considerada como a

representação mais fiel do cenário de trânsito. Posteriormente é apresentado o serviço de sentimento das rotas do T-MAPS, com o objetivo de descrever com mais detalhes os trajetos fornecidos pelo *Google Directions*.

Os resultados apresentados são referentes à segmentação da região de Manhattan em Nova Iorque. Após as análises de frequência de *tweets* em Manhattan, notou-se que a cobertura espacial e temporal dos dados viabiliza o uso das subdivisões do mapa. No nível de bairros (existem 29 bairros oficiais em Manhattan[4]), todos os bairros apresentam *tweets* nas faixas de tempo analisadas. Seguindo a modelagem descrita anteriormente, obtivemos grafos para cada estratégia de custo (veja Seção 4). Os grafos contêm 29 vértices (bairros) e arestas entre os bairros adjacentes.

5.1. Serviço de rotas T-MAPS

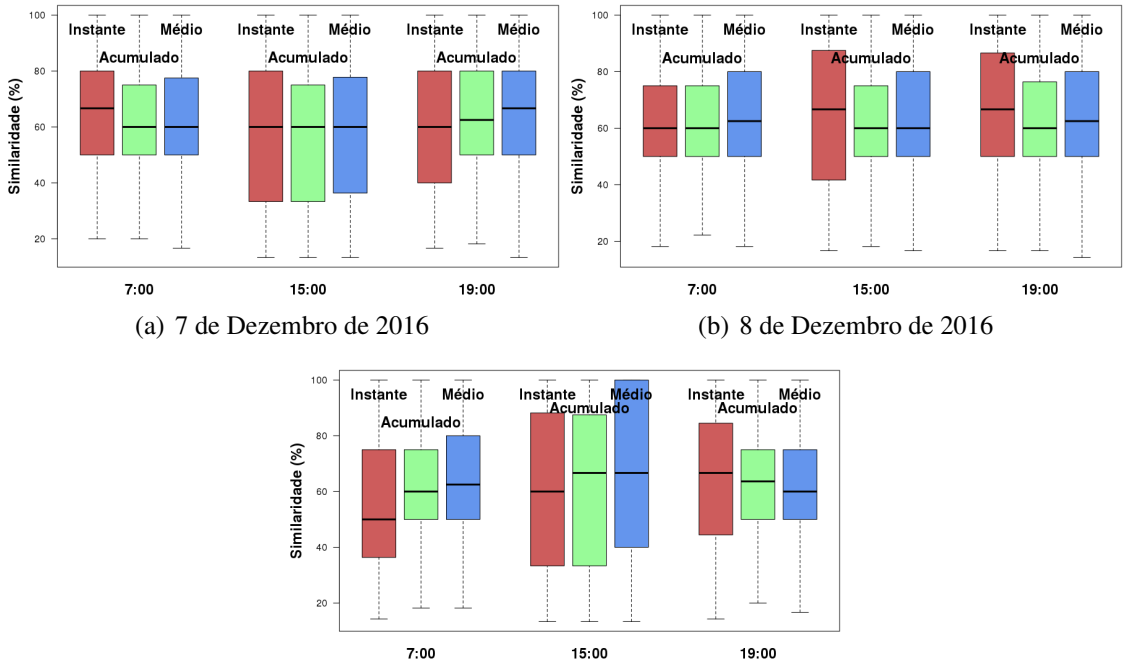
Este serviço do T-Maps sugere rotas considerando uma visão macro entre regiões no mapa, ou seja, o objetivo é dizer ao usuário que as regiões recomendadas possuem as melhores condições segundo a métrica aplicada. A avaliação deste serviço foi realizada comparando as regiões percorridas pelas rotas providas pelo T-MAPS[5] e *Google Directions*. A comparação entre os trajetos foi baseada na similaridade entre as regiões sugeridas, ou seja, buscou-se identificar o percentual de similaridades da abordagem T-MAPS em relação ao *Google Directions*. Foram geradas 812 rotas a partir da combinação de cada um dos 29 bairros (como origens e destinos) de Manhattan, exceto aqueles em que a origem e destino são os mesmos. Além disso, rotas de $A \rightarrow B$ e de $B \rightarrow A$ também são consideradas. Essas rotas foram coletas do T-MAPS e *Google Directions* em três momentos do dia (7, 15 e 19 horas) escolhidos propositalmente, pois, geralmente, são os horários onde as ocorrências de *tweets* e congestionamentos são mais expressivos.

As similaridade é definida como o percentual de interseção das rotas recomendadas pelo T-MAPS e *Google Directions*. A Figura 8 apresenta uma amostra da avaliação de similaridade em três dias de coleta para três métricas de custo do T-MAPS, totalizando 21924 rotas analisadas. É possível notar que, a métrica *Instante* resulta na maior variação da mediana da taxa de similaridade, entre 50% e 66%, enquanto as métricas *Acumulado* e *Médio* apresentam variação entre aproximadamente 60% e 66%. Interpreta-se que na métrica *Instante*, ao menos a metade das rotas consideradas possuem 50% de similaridade com as rotas do *Google Directions*, ao passo que a métrica *Acumulado* e *Médio* apresentam ao menos 60% de similaridade. Também é importante ressaltar que, na média de todos os dias e horários avaliados, metade das rotas apresentaram ao menos 62% de similaridade entre T-MAPS e *Google Directions*. Além disso, uma porcentagem relevante de 25% das rotas apresentou grau de similaridade entre 87% e 100%. Finalmente, ressalta-se que 75% das rotas avaliadas apresentam grau de similaridade superior a 47%

Portanto, uma modelagem que descreve o cenário de trânsito, baseada exclusivamente nos *tweets*, fornece significativo grau de correspondência nos cenários avaliados, quando comparado com a ferramenta *Google Directions*, que por sua vez baseia-se em diversas fontes de dados. Em consequência desta similaridade, o T-MAPS pode contribuir com o enriquecimento do cenário de trânsito fornecendo contexto, baseado nos textos dos *tweets*. Um dos possíveis serviços de enriquecimento das rotas é a análise dos sentimentos

[4] `www1.nyc.gov/site/planning/data-maps/open-data/districts-download-metadata.page`

[5] Foi usado o algoritmo de Dijkstra sobre o grafo ponderado para computar o menor caminho.



(a) 7 de Dezembro de 2016

(b) 8 de Dezembro de 2016

(c) 9 de Dezembro de 2016

Figura 8. Similaridade entre rotas sugeridas pelo T-MAPS e *Google Directions*

de um trajeto, o qual será explorado na seção seguinte.

5.2. Serviço de sentimento da rota

Esta seção descreve uma possível técnica para prover o serviço de sentimento de rota usando o T-MAPS. Para isso, realizou-se um pré-processamento sobre o texto de todos os dados coletados. Inicialmente foram removidas todas as palavras de parada (*Stop Words*), *links*, caracteres especiais e pontuação. Em seguida, executou-se um processo chamado *stemming* sobre o texto. O *stemming* remove sufixos das palavras como, por exemplo, *"Jaming"*, *"Jammed"* para que todas sejam consideradas como apenas *Jam*. Como resultado, gerou-se a nuvem de palavras exibida na Figura 9(a), em que o tamanho da palavra indica a maior frequência de uso nos *tweets*.

Para extrair o sentimento do texto dos *tweets*, foi utilizado o pacote de acesso livre *syuzhet*[6] da ferramenta R. Esse pacote contém algoritmos para extração de emoções e sentimentos de textos. A ideia dos algoritmos é utilizar dicionários contendo várias palavras e emoções/sentimentos associados a elas. A definição da emoção ou sentimento de um determinado texto depende do número de ocorrências dessas palavras no texto, ou seja, verificam-se as palavras mais comumente utilizadas e o grau de emoção/sentimento é atribuído ao texto.

A Figura 9(b) apresenta o resultado da análise de sentimento no mapa e uma rota sugerida pelo *Google Directions*. Com essas duas informações (rota e sentimento), o T-MAPS pode enriquecer o trajeto sugerido, indicando aos usuários o sentimento das regiões do trajeto ou até mesmo prover o roteamento baseado nestes sentimentos.

[6] `https://cran.r-project.org/web/packages/syuzhet/syuzhet.pdf`

(a) Palavras frequentes nos *tweets* (b) Manhattan dia 14 de Out

Figura 9. Análise de sentimentos sobre os texto dos *tweets* coletados

6. Trabalhos Relacionados

Diversos trabalhos com o intuito de analisar os sentimentos de determinada região, ou ainda, propor recomendações de locais baseadas em informações obtidas de LBSMs, são encontrados na literatura [Bertrand et al. 2013, Kim et al. 2014]. Contudo, de forma diferente dos trabalhos citados, o estudo e a modelagem aqui apresentados propõem associar dados de LBSMs, em especial do Twitter, às regiões do mapa de interesse e, então, enriquecer o contexto de navegação atual como, por exemplo, vinculando os sentimentos dos *tweets* às rotas geradas por plataformas tais como o *Google Maps*.

Na difícil tarefa de explorar o texto dos dados, alguns trabalhos têm obtido alguns avanços que podem aperfeiçoar o T-MAPS. Em [Ribeiro Jr et al. 2012], os autores apresentam um sistema para detectar e localizar eventos usando o texto de *tweets*. Em [Gong et al. 2015], é apresentado um sistema que identifica e sinaliza aos usuários pontos de congestionamento utilizando LBSMs. Em relação a estudos do comportamento, mobilidade, trajetórias e demandas de viagens diversos trabalhos têm recebido destaque [Yin and Du 2016, Kung et al. 2014, Cho et al. 2011, Zheng et al. 2008]. Esses trabalhos apresentam de um modo ou de outro uma complementação à abordagem apresentada neste trabalho, visando compreender melhor o cenário do trânsito como um todo.

7. Conclusão e Trabalhos Futuros

Este trabalho apresentou um estudo de caracterização e relacionamento entre dados de plataformas LBSMs e o cenário de trânsito. Além disso, foi apresentado o T-MAPS que usa um modelo baseado em grafo e dados do Twitter, com o objetivo de enriquecer o contexto de navegação na cidade, associando informações dos *tweets* às rotas geradas. Na avaliação dos serviços providos pelo T-MAPS, mostrou-se que as rotas sugeridas apresentam em média 62% de similaridade com as do *Google Directions* e em 25% das rotas avaliadas foram obtidos graus de similaridade entre 87% e 100%. Isso indica que o T-MAPS fornece significativa correspondência com as rotas do *Google Directions*, mesmo utilizando exclusivamente *tweets*. Também foi apresentado um serviço de enriquecimento das rotas sugeridas pelo *Google Directinos* baseado no sentimento das regiões, por sua vez extraídos do processamento dos textos dos *tweets*.

Finalmente, pretende-se como trabalhos futuros estender o T-MAPS para regiões com maiores dimensões, explorar os textos dos *tweets* a fim de se extrair mais informações sobre os eventos reportados. Além disso, pretendemos avançar em questões relacionadas à confiabilidade das fontes e validade dos dados.

Referências

[Bazzan and Klügl 2013] Bazzan, A. L. and Klügl, F. (2013). Introduction to intelligent systems in traffic and transportation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 7(3):1–137.

[Bertrand et al. 2013] Bertrand, K. Z., Bialik, M., Virdee, K., Gros, A., and Bar-Yam, Y. (2013). Sentiment in new york city: A high resolution spatial and temporal view. *arXiv preprint arXiv:1308.5010*.

[Cho et al. 2011] Cho, E., Myers, S. A., and Leskovec, J. (2011). Friendship and mobility: user movement in location-based social networks. In *17th ACM SIGKDD*, pages 1082–1090. ACM.

[Florea et al. 2009] Florea, M. C., Jousselme, A.-L., Bossé, É., and Grenier, D. (2009). Robust combination rules for evidence theory. *Information Fusion*, 10(2):183–197.

[Gong et al. 2015] Gong, Y., Deng, F., and Sinnott, R. O. (2015). Identification of (near) real-time traffic congestion in the cities of australia through twitter. In *ACM Understanding the City with Urban Informatics*.

[Kim et al. 2014] Kim, J., Cha, M., and Sandholm, T. (2014). Socroutes: safe routes based on tweet sentiments. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 179–182. ACM.

[Kung et al. 2014] Kung, K. S., Greco, K., Sobolevsky, S., and Ratti, C. (2014). Exploring universal patterns in human home-work commuting from mobile phone data. *PloS one*, 9(6):e96180.

[Li and Sun 2014] Li, C. and Sun, A. (2014). Fine-grained location extraction from tweets with temporal awareness. In *37th ACM SIGIR*, pages 43–52. ACM.

[Liu et al. 2011] Liu, X., Zhang, S., Wei, F., and Zhou, M. (2011). Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*.

[Rettore et al. 2016] Rettore, P. H., Santos, B. P., Campolina, A. B., Villas, L. A., and Loureiro, A. A. (2016). Towards intra-vehicular sensor data fusion. *19th International Conference on ITS*.

[Ribeiro Jr et al. 2012] Ribeiro Jr, S. S., Davis Jr, C. A., Oliveira, D. R. R., and Meira Jr, W. (2012). Traffic observatory: a system to detect and locate traffic events and conditions using twitter. In *5th ACM SIGSPATIAL*.

[Yin and Du 2016] Yin, J. and Du, Z. (2016). Exploring multi-scale spatiotemporal twitter user mobility patterns with a visual-analytics approach. *ISPRS International Journal of Geo-Information*, 5(10):187.

[Zadeh 1984] Zadeh, L. A. (1984). Review of a mathematical theory of evidence. *AI Magazine*.

[Zheng et al. 2008] Zheng, Y., Li, Q., Chen, Y., Xie, X., and Ma, W.-Y. (2008). Understanding mobility based on gps data. In *10th ACM Ubiquitous computing*.

Análise de resposta em frequência para modelagem e geração de carga de trabalho em aplicações de *e-commerce*

Lourenço Alves Pereira Júnior[1,2], Edwin Luis Choquehuanca Mamani[1], Regina H. Carlucci Sanatana[1], Marcos José Santana[1], Francisco José Monaco[1]

[1]Instituto Federal de São Paulo — IFSP
ljr@ifsp.edu.br

[2]Universidade de São Paulo — USP
{ljr,edwin,rcs,mjs,monaco}@icmc.usp.br

Abstract. *This paper aims to evaluate the performance of an e-commerce system under a time-varying workload. The approach is based on the Transfer Function Model which enables analysis in the frequency-domain allowing new insigths towards workload generation. The main contributions include the novel approach for performance modeling of computing systems and the system's model as parameter for the workload generation.*

Resumo. *Este artigo apresenta a avaliação de desempenho de um sistema de* e-commerce *operando em condições de carga de trabalho variantes no tempo. A abordagem é baseada em Função de Transferência, permitindo a análise no domínio da frequência. As principais contribuições incluem a abordagem inovadora para modelagem de desempenho e a geração de carga que considera o desempenho como parâmetro.*

1. Introdução

A computação em nuvem é uma realidade no ciclo de vida de desenvolvimento de sistemas computacionais atuais. Dadas suas características essenciais [Mell and Grance 2011], a potência computacional disponível para uma aplicação (capacidade) é basicamente variável (elasticidade), e a demanda tende a ser flutuante, sendo influenciada por sazonalidades e padrões de acessos, promoções eventuais (*e.g: black friday*) [Yuan et al. 2013, Santos et al. 2016, Silva et al. 2013]. Nesse sentido, a modelagem do desempenho *dinâmico* é importante pois permite que a capacidade computacional seja alterada em tempo de execução em função da carga de trabalho [Mamani et al. 2015].

No contexto de aplicações Web, observa-se a adoção de sistemas de computação em nuvem como plataforma execução [Qu et al. 2016]. A carga de trabalho variante no tempo é um fator importante para sistemas de *e-commerce*, pois percas de desempenho diminuem a taxa de conversão dos clientes, o que ocasiona diminuição de vendas. A variação da carga traduz-se como um elemento que perturba o regime estacionário de um sistema, gerando um comportamento transiente antes do eventual novo regime estacionário. É durante o transiente que sistemas experimentam condições extremas de sua utilização e gargalos são revelados [Veeraraghavan et al. 2016].

Quando relaciona-se a demanda e a capacidade de um serviço em nuvem para modelagem do desempenho, técnicas tradicionais (como na modelagem baseada em Teoria de Filas) apresentam-se desafiadoras dada sua natureza estacionária [Yang and Liu 2012].

Nesse cenário, um requisito importante para o planejamento de capacidade *dinâmica* é entender como o sistema reage a diferentes pertubações na carga de trabalho. Este artigo (1) apresenta uma abordagem de modelagem de desempenho não estacionária baseada em Função de Transferência e (2) analisa o comportamento de um sistema sujeito a diferentes variações na carga de trabalho. As contribuições deste trabalho são a análise de resposta em frequência do desempenho de um sistema de *e-commerce* real e a aplicação do método para a geração de carga de trabalho (com rajadas e baseada em traços). O diferencial de nossa proposta é considerar o modelo do sistema como parâmetro no processo de geração.

A seguir, na Seção 2 são discutidos trabalhos relacionados. A Seção 3 apresenta o sistema em estudo e sua modelagem. A Seção 4 apresenta a análise de resposta em frequência. Nas Seções 5 e 6 aplicam-se os conceitos abordados para a especificação de dois métodos para geração de carga de trabalho. Por fim, na Seção 7 conclui-se este estudo.

2. Trabalhos Relacionados

A avaliação de desempenho tradicional utiliza de modelos capazes de predizer o desempenho de sistemas em regime estacionário [Jain 1990]. No entanto, conforme apresentado por [Yang and Liu 2012, Pereira 2016], há um esforço para representação analítica do comportamento transiente. Um exemplo da importância da modelagem do transiente pode ser observado em ambiente de computação em nuvem em que flutuações na demanda podem fazer com que Máquinas Virtuais sejam ligadas desnecessariamente [da Luz et al. 2016]. [Hellerstein et al. 2004] apresentam métodos e técnicas para esse fim por meio da abordagem de Função de Transferência (FT).

Com base nesses estudos e na aplicação desses modelos para a avaliação de desempenho transiente, [Pereira et al. 2015b] propuseram um modelo conceitual de requisitos denominado *Monitor*, *Effector*, *Demand* e *Capacity* (MEDC). Sua finalidade é produzir um arcabouço para instrumentação e/ou adaptação de sistemas computacionais para a execução adequada de experimentos para esse tipo de avaliação de desempenho. O modelo conceitual possibilita a modulação da demanda e da capacidade, baseada no monitoramento periódico das métricas de desempenho de interesse e atuações específicas para o sistema em avaliação [Pereira et al. 2015a].

Neste artigo, essa abordagem para avaliação de desempenho é aplicada a um sistema de *e-commerce* baseado no *benchmark* Bench4Q [Zhang et al. 2011]. Seu projeto contempla a análise e produção de resultados estacionários como uma ferramenta orientada a qualidade de serviço (QoS). Trata-se da extensão do TPC-W (*Transaction Processing Performance Council — transational web benchmark*) [Menasce 2002]. As principais melhorias do Bench4Q incluem o suporte a métricas baseadas em sessões (não apenas em requisições) e a geração de carga de trabalho sensível a QoS para fins de planejamento de capacidade [Zhang et al. 2011]. [Mamani 2016] estendeu o Bench4Q, transformando-o em uma ferramenta de *benchmarks* para avaliação de desempenho não estacionária e [Souza 2016] implementou o módulo gerador de carga de trabalho (*Demand* do modelo MEDC) para que fosse possível a geração de cargas variantes no tempo.

Como um dos desafios apresentados por [Huang et al. 2014] está a adequação de modelos dinâmicos aplicados à implementação de Teoria de Controle com aplicação em gerência de recursos. Isso é importante, pois a avaliação é feita de modo

offline com uma carga capaz de expor o comportamento dinâmico do sistema. Avaliações do tipo *what-if* são importantes [Jiang et al. 2016], mas tem-se o ônus de várias execuções e configurações para obter resultados úteis. Modelos autor-regressivos possuem características recursivas e requerem baixo custo computacional [Yang and Liu 2012]. Portanto, há uma tendência de aprimoramento na utilização da Teoria de Controle para o gerenciamento de recursos em ambientes de computação em nuvem [Papadopoulos et al. 2015]. O que motiva a necessidade de diferenciação de serviço em aplicações de Internet multi-camadas com desempenho controlado [Diao et al. 2006].

Esses trabalhos têm em comum a abordagem: (1) baseada em FT para lidar com a carga de trabalho variante no tempo, algo comum no domínio de computação em nuvem e (2) *ad hoc* para resolução de problemas. Percebe-se a falta de iniciativas para especificação de um método genérico para modelagem de desempenho de aplicações de três camadas em ambiente de nuvem, bem como sua análise de resposta em frequência e geração de carga de trabalho que levem em conta seu comportamento dinâmico.

3. Sistema em Estudo

O sistema em estudo é uma aplicação web de três camadas, uma loja virtual (*e-commerce*) parte do Bench4Q [Zhang et al. 2011] . Foram feitas as adaptações necessárias para que esse *benchmark* se adequasse ao modelo conceitual MEDC [Pereira et al. 2015a].

A Figura 1 ilustra a implementação, as interações entre os componentes e a forma de implantação (*deploy*). O nome dado a esta nova versão é OnlineBench4Q, pois é possível alterar as características da carga de trabalho em tempo de execução (e.g: requisições com médias variantes no tempo) e capacidade adaptativa (*e.g: auto scaling*). Na *camada de apresentação*, as requisições são feitas pelos *emulated browsers* (EBs). Antes da execução um arquivo de configuração é passado para o módulo *Demand* para especificar como a carga de trabalho será gerada durante o experimento. Existem três formas básicas de especificação: (1) **carga sintética**, realizada através de uma distribuição de probabilidade (exponencial, normal etc.) e (2) **traço**, um arquivo com a lista de requisições a serem executadas.A *camada de aplicação* é composta por um balanceador de carga implementado pelo HAProxy[1] — que fornece a métrica de quantas requisições por segundo chegam à aplicação — que recebe as requisições e as repassa para os servidores virtualizados (round-robin). Caso a taxa de utilização seja alta/baixa, adiciona-se/remove-se novas VMs. O mecanismo de gerenciamento de recursos é conforme especificado em [Mamani et al. 2015, Pereira et al. 2015a]. O `Monitor` afere a taxa de utilização a cada segundo através do `proc` do Linux. O `Capacity` é implementado por um controlador Proporcional-Integral. O `Effector` executa alterações nas máquinas virtuais (VMs) dessa camada. A *camada de dados* foi implementada pelo PostgreSQL. A Tabela 1 detalha o sistema físico para execução dos experimentos.

O modelo de desempenho adotado relaciona a quantidade de requisições por segundo (req/s) e a média da taxa de utilização dos servidores de aplicação. A carga de trabalho é gerada por *browsers* emulados (*emulated browsers* — EBs) em modo aberto, com sessões configuradas para manter uma média de 47 req/s por EB, e o grafo CBMG arranjado para acesso prioritário a páginas que demandam operações de leitura na camada de dados. O fluxo de requisição de todos EBs é combinado no HAProxy. Foi possível

[1] Sítio do HAProxy: `http://www.haproxy.org/`

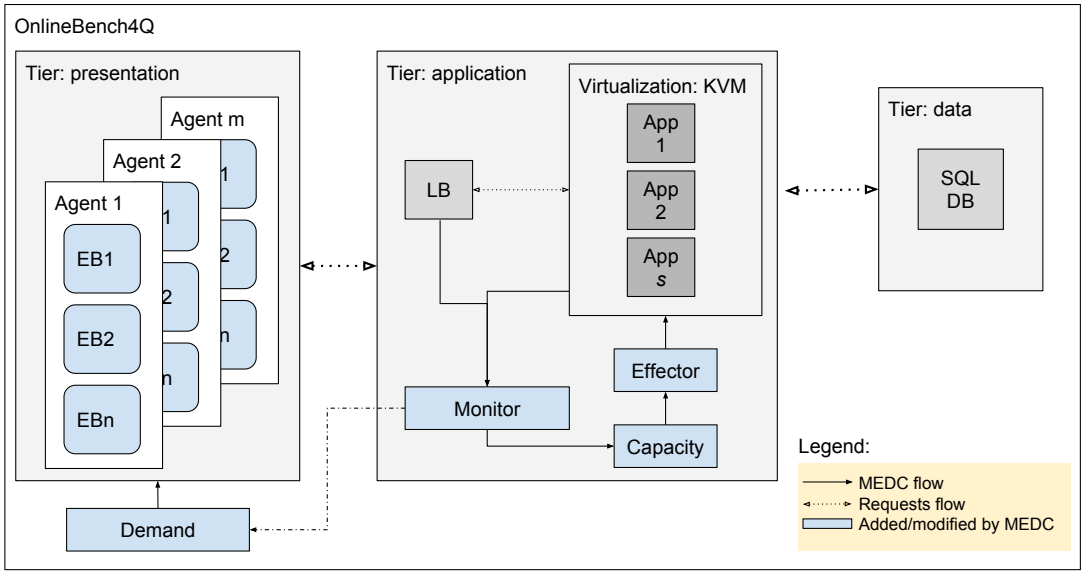


Figura 1. Diagrama arquitetural de componentes do OnlineBench4Q, uma combinação de Bench4Q [Zhang et al. 2011] e MEDC [Pereira et al. 2015a].

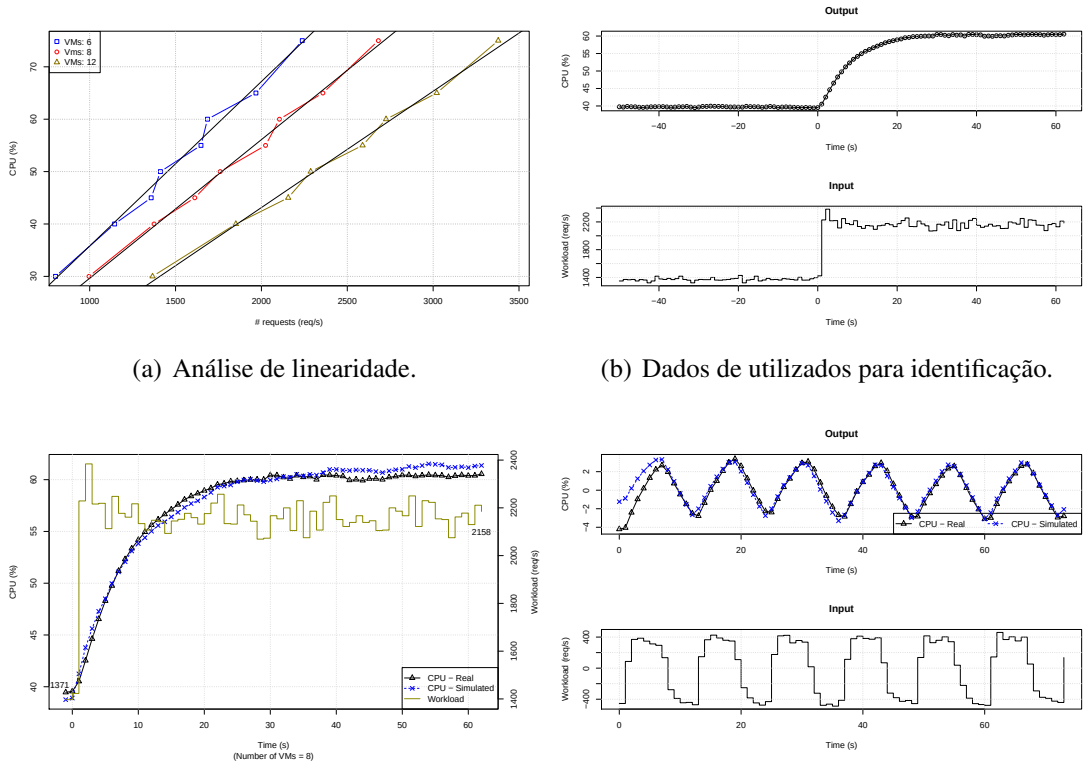
observar que, para esse cenário, quanto maior a quantidade de requisições, maior a taxa de utilização, caracterizando-se como uma relação linear entre essas duas variáveis — comprovada com os experimentos da Seção 3.1. Tal relação linear determina a região de operação desse sistema em conformidade com a carga avaliada. Um parâmetro para o sistema que altera esse comportamento é a quantidade de VMs. Mesmo que a taxa req/s seja constante, ao adicionar VMs, a taxa de utilização diminuirá. Convenciona-se como W a entrada para esse sistema (req/s), U para a saída (taxa de utilização média dos servidores de aplicação) e VMs a perturbação. A Seção 3.1 apresenta procedimentos com o propósito de quantificar o comportamento dinâmico entre as variáveis deste sistema.

3.1. Identificação do Sistema

A identificação de sistema é uma área de pesquisa que estipula métodos para execução de experimentos e análise de sistemas [Ljung 1999]. O processo de identificação adotado consiste em: (1) determinação do sistema: especificação de variáveis com relação de causa e efeito; (2) determinação do escopo: definição do tempo de amostragem e do gerador de carga de trabalho; (3) excitação do sistema: determinação de um sinal de entrada capaz de expor a dinâmica do sistema; (4) identificação do modelo: execução

Tabela 1. Configuração do hardware utilizado nos experimentos.

Nome	CPU (GHz)	Mem. (GB)	HD (Mb/s)	SO	#
Load Balancer	INTEL i5-3330-(3.0) x4	7,8	87,74	ClearOS 6.6	1
DB server (pool)	AMD Q6600-(2.4) x4	7,8	71,21	Ubuntu 14.04.2	8
DB server (master)	FX-8320-(3.5) x8	23	285,58	Ubuntu 14.04.3	1
Clientes	AMD Q6600-(2.4) x4	7,8	76,96	Ubuntu 14.04.2	9
Hipervisor	X5660-(2.8) x12	11	251,98	Ubuntu 15.04	1
Servidor de VMs	X5660-(2.8) x12	11	251,98	Ubuntu 15.04	4
VMs	QEMU Virtual x1	1	178,60	Ubuntu 12.04.5	-



(a) Análise de linearidade.

(b) Dados de utilizados para identificação.

(c) Aderência aos dados empíricos.

(d) Validação com sinal onda quadrada.

Figura 2. Resultados obtidos do processo de identificação do sistema.

de experimentos para gerar dados e parametrização do modelo; (5) verificação: métricas estatísticas para avaliar o modelo; (6) validação: acurácia do modelo em relação outros sinais de entrada. Esses passos são detalhados em [Ljung 1999, Hellerstein et al. 2004, Pereira 2016]. A Figura 2 apresenta os resultados obtidos do processo de identificação.

Os passos 1 e 2 da identificação foram contemplados pela relação de linearidade entre as variáveis e é apresentada na Figura 2(a). Para a geração desses dados configurou-se experimentos da seguinte forma: variou-se a quantidade de VMs, valores $\{6, 8, 12\}$. Para cada quantidade de VMs encontrou-se uma taxa de req/s estacionária tal que impactasse na variável U nos patamares de $\{30, 40, 50, 60\}$. Dessa forma, para cada combinação de VMs e U aplicou-se a carga W (req/s) correspondente, aumentando U em 15% no meio do experimento. Mediu-se W e U periodicamente a uma taxa de amostragem de 1 s. Descartou-se os dados do período não estacionário. Em suma, conclui-se que o sistema é linear. Para este trabalho fixou-se a quantidade de VMs em 8 (oito) e a região de operação na faixa em que W está entre 1.371 e 2.158 req/s, fazendo com que U opere na faixa entre 40% e 60%, respectivamente. O ponto de operação é $\bar{U} = 50\%$.

O sinal de excitação adotado foi o de degrau (passo 3). Um sinal em degrau significa que o sistema deve estar em um ponto de operação com uma carga de trabalho estacionária e em determinado instante alterar abruptamente essa carga de modo que o sistema entre em outro patamar estacionário. Uma vantagem da utilização deste sinal é sua simplicidade de implementação e execução, sendo capaz de expor a dinâmica de um sistema computacional linear. A Figura 2(b) apresenta os resultados. W é mantida a uma taxa estacionária de 1.351 req/s e, no instante de tempo 0 (zero), houve uma mudança

abrupta para 2.158 req/s. Observa-se o comportamento dinâmico entre as duas variáveis, pois U muda de 40% para 60% não instantaneamente, mas sim gradativamente, semelhante a um modelo exponencial (i.e: $U = g \cdot (1 - e^{-\lambda}) \cdot W$), em que g é o ganho e λ define a velocidade em que o novo patamar estacionário será atingido).

O modelo utilizado para a identificação foi o Autorregressivo com entradas exógenas (ARX). Características importantes desse modelo incluem: sua flexibilidade de parametrização, linearidade, discreto e capacidade de representação da estocasticidade. A Figura 2(c) apresenta uma simulação do modelo identificado em relação aos dados empíricos (passo 4). O sistema identificado é descrito como um modelo ARX de primeira ordem com os parâmetros estimados $a_1 = -0.9037$ e $b_1 = 0.00273$, que pode ser representado em formato de Função de Transferência como

$$F(z) = \frac{b}{1 + az^{-1}} = \frac{0.00273}{1 - 0.9037z^{-1}}. \quad (1)$$

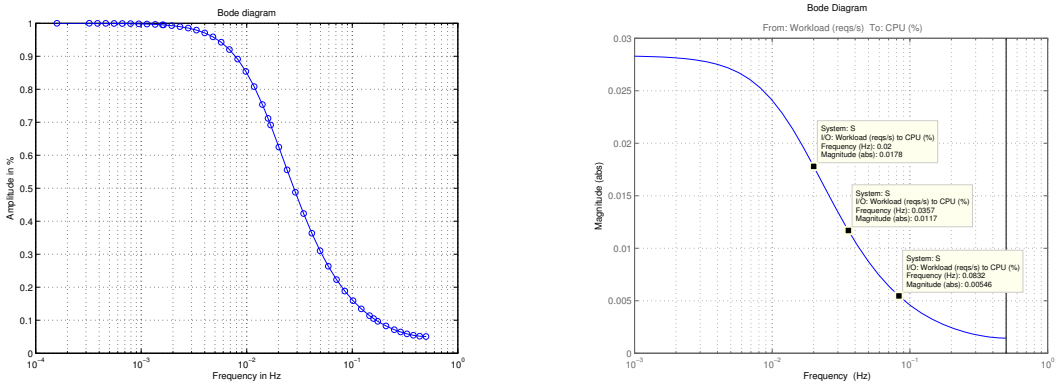
Os valores para RMSE e R^2 foram altos e satisfatórios (passo 5), $0,7924$ e $0,8661$ respectivamente. A Figura 2(d) apresenta a validação de $F(z)$ capaz de representar o comportamento do sistema na faixa de operação (passo 6).

4. Análise de Resposta em Frequência

O desempenho do sistema foi modelado como um sistema dinâmico linear e invariante no tempo (*Linear Time-invariant* — LTI) e representado por uma Função de Transferência de primeira ordem (Eq. 1). Isso significa que, respeitada a região de operação, pode-se simular o comportamento dinâmico do desempenho de um sistema face a diferentes cargas de trabalhos. Isso é um ponto importante, pois permite que resultados sejam produzidos por simulação, em vez de experimentos que acarretam em custos. Dada a simplicidade e a característica recursiva de um modelo de FT, os resultados são praticamente instantâneos [Yang and Liu 2012]. Um método a ser explorado nesse tipo de modelagem de desempenho é a resposta em frequência, representado pelo diagrama de Bode.

O diagrama de Bode é um modelo não paramétrico em que utiliza-se a frequência para predição de desempenho (Figura 3). O eixo x representa a frequência com que eventos ocorrem, e no y o quanto do desempenho estacionário será atingido. O gráfico é produzido, teoricamente, ao excitar o sistema com uma onda senoidal de frequência x, obtendo a amplitude do sinal de saída y. A leitura do gráfico de resposta em frequência é feita ao interpretar o quanto a variável de saída é atenuada em relação a variações na variável de entrada. Por exemplo, seja uma onda quadrada (Figura 4(a)) de período P segundos e amplitude A. Se P tiver comprimento $0,5$ s, a frequência será 2 Hz; se tiver 1 s, frequência de 1 Hz; 2 s, frequência de $0,5$ Hz; e assim por diante. Dessa forma, define-se frequência como $f = 1/P$. Deve-se ressaltar que o período da onda quadrada possui dois patamares, cada um correspondendo a metade de P, $D = P/2$. Na Figura 3(a), se $P = 32$ s, $D = 16$ s e $f = 0,003$ Hz ($3 \cdot 10^{-2}$), haverá uma atenuação de aproximadamente 50% do valor final atingido em estacionário. Portanto, para o sistema $F(z)$, Eq. 1, o impacto no desempenho do sistema é o inverso da frequência com que os eventos ocorrem.

O diagrama Bode serve como referência para quantificação da relação entre frequência e impacto no desempenho (Figura 3). Eventos de alta frequência para o sistema em estudo neste artigo caracterizam como alteração abruptas nos valores, porém com



(a) Diagrama de Bode de atenuação relativa.

(b) Diagrama de Bode.

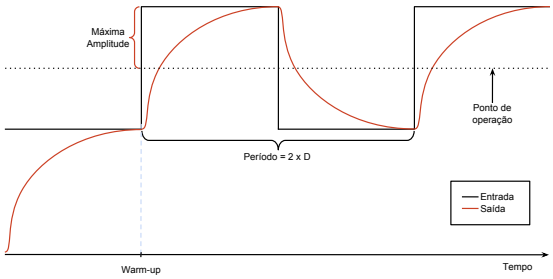
Figura 3. Resposta em frequência do sistema $F(z)$, Eq. 1.

duração baixa incapaz de influenciar no desempenho. Seja W em regime estacionário e dados coletados de execução única — inspecionar Figura 6(b). O sinal é ruidoso pois observa-se que há um patamar médio de req/s e uma variação em torno desta média. A representação desse sinal no domínio da frequência é composta por uma frequência 0 Hz sobreposta de outras frequências mais altas. A frequência zero corresponde ao valor médio de W. Uma conclusão importante é que esse ruído influenciará pouco na variável de saída do sistema identificado (Eq. 1). Os picos e vales causados pela alta frequência se anulam, caso em que há uma variância maior na entrada do que na saída.

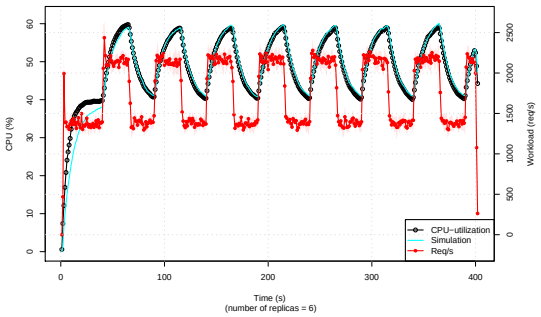
Como prova de conceito, foram escolhidas 3 (três) frequências de interesse a fim de analisar se os valores previstos pelo diagrama de Bode representam o desempenho do sistema. A Figura 3(b) apresenta as frequências anotadas e os valores absolutos de atenuação. Os experimentos foram conduzidos aplicando-se $W = 1.371$ por 40 s, correspondente ao tempo transiente inicial (*warm-up*), seguidos de uma excitação do tipo onda quadrada, com duração de $\{6, 14, 25\}$ s. A duração, D_i, de uma frequência, f_i, pode ser obtida pelo cálculo $D_i = 1/2f_i$. A atenuação em U é observada pela máxima amplitude realizada no experimento e dada por $A = mag/g$, em que mag é magnitude, observada na Figura 3(b), e g é o ganho do sistema, neste caso $0,0283$. A terminologia adotada é representada na Figura 4(a). Cada experimento foi replicado 6 (seis) vezes e o intervalo de confiança de 95% foi calculado. A Figura 4 apresenta os resultados.

O mesmo sinal de entrada W produzido pelo sistema real foi passado para a FT, resultando em um sinal de saída simulado. Ou seja, o sinal de entrada foi passado tanto para o sistema real quanto para a FT. Observa-se que o modelo de FT para representação do desempenho é adequado. Cada experimento possui o valor médio de $\bar{U} = 50\%$, com o sinal U excursionando nas amplitudes mínimas e máximas, patamares alto e baixo. A Tabela 2 apresenta a acurácia dos resultados obtidos.

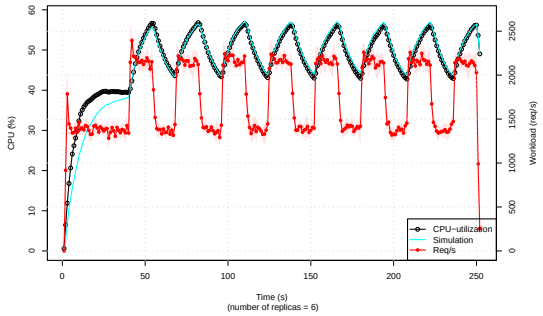
A atenuação de frequências mais altas podem ser observadas na Figura 5. Neste caso obteve-se o valor de W estacionário e com duração de 512 s a partir de um experimento no sistema físico. A transformada de Fourier permite observar esse sinal no domínio da frequência. Utilizou-se a transformada de Fourier rápida (FFT) para este fim. O resultado é o espectro do sinal, isto é, o conjunto de frequências que compõem aquele sinal. Escolheu-se três frequências com a finalidade de observar como elas são atenua-



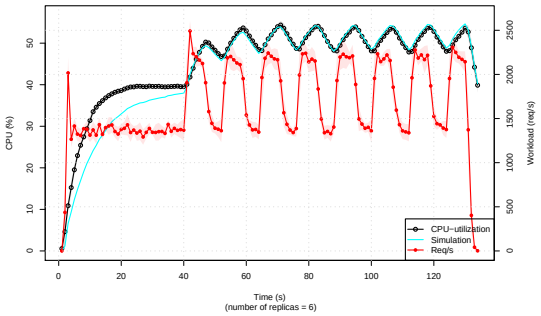
(a) Terminologia adotada para os experimentos.



(b) Resposta em frequência com duração de 25 s.



(c) Resposta em frequência com duração de 14 s.



(d) Resposta em frequência com duração de 6 s.

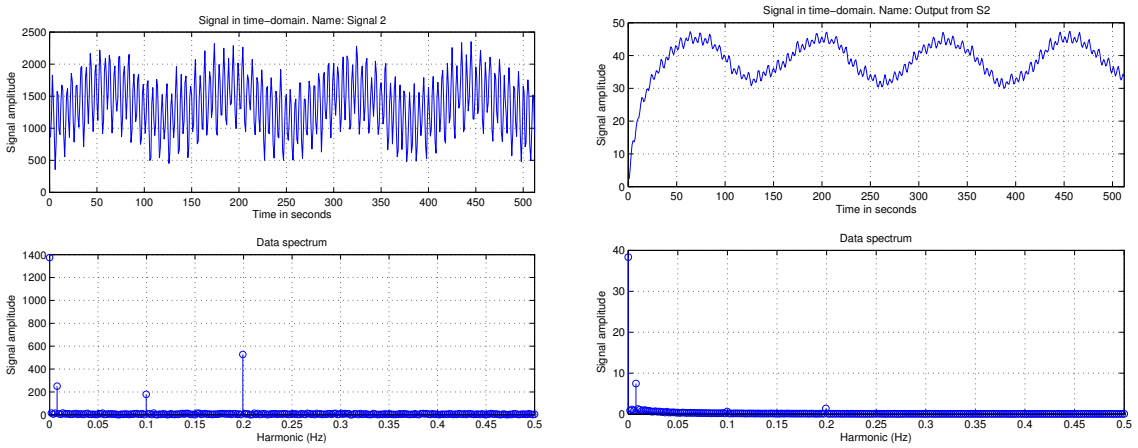
Figura 4. Resposta em frequência. Ponto de operação a 50% de CPU. Denomina-se o sinal como onda quadrada ou trem de pulsos por seu formato característico.

das após passar pelo sistema. As frequências escolhidas foram $\{0.0075, 0.01, 0.02\}$ Hz. Assim, após a obtenção do espectro, multiplicou-se cada frequência por $\{1.2, 1.4, 1.6\}$ respectivamente, amplificando-as. O sinal desejado foi computado a partir da transformada inversa de Fourier. Em suma, obteve-se o sinal no domínio do tempo; levou-o para o domínio da frequência; alterou-se algumas frequências; o trouxe para o domínio do tempo novamente. A Figura 5(a) apresenta o sinal após essa alteração (o sinal antes da alteração é apresentado na Figura 6(b)). A frequência alterada mais baixa, 0.0075, é responsável pela oscilação mais observável, e as outras duas frequências correspondem às oscilações de menor duração e com amplitudes contundentes.

Ao passar esse novo sinal de entrada, W', pela FT, obtém-se U conforme observado na Figura 5(b). Ao inspecionar seu espectro, as frequências adicionadas mais

Tabela 2. Acurácia dos valores baixo e alto preditos pelo modelo.

Duração Degrau	Predito (Pr) (CPU %)	Empírico (Em) (CPU %)	Resíduo (Pr - Em)
25 alto	56,27	59,095	-2,825
25 baixo	43,73	40,305	3,425
14 alto	54,12	56,372	-2,252
14 baixo	45,88	43,099	2,781
6 alto	51,92	53,406	-1,486
6 baixo	48,08	47,960	0,12



(a) Entrada. (b) Saída.

Figura 5. Comportamento de atenuação de frequências.

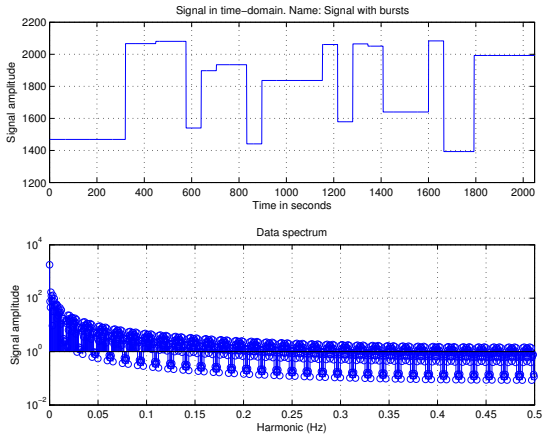
altas, 0.01 e 0.02, foram consideravelmente mais atenuadas se comparadas à frequência mais baixa, 0.0075. Mas todas as frequências mais altas (localizadas à direita) foram atenuadas também. As frequências mais baixas (à esquerda) foram menos atenuadas. A quantificação dessa atenuação é a mesma descrita pelo diagrama de Bode (Figura 3).

5. Geração de Rajadas

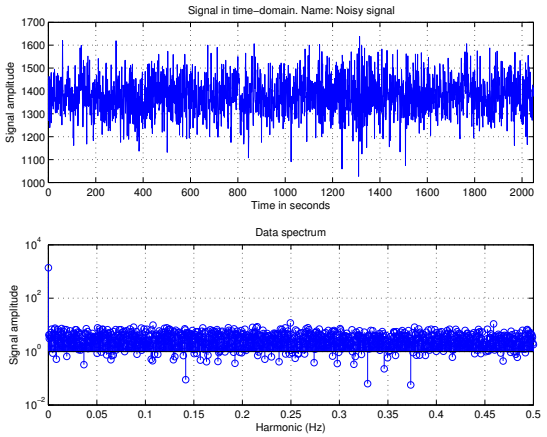
Uma aplicação que pode ser feita a partir da FT e da análise em frequência é a especificação de um método para a geração de carga de trabalho com rajadas. Uma característica importante dessa abordagem é considerar a carga de trabalho que relaciona o desempenho do sistema, um método diferente daqueles comumente adotados [Centurion et al. 2012]. Devido ao fato de ser um modelo gerado por meio de dados empíricos, pode-se parametrizar como a carga irá impactar no sistema de forma precisa. Seja a especificação de uma rajada tal que haja restrições nos períodos de flutuações no número de req/s, não tão curtas de modo que elas sejam absorvidas pela inércia do sistema, nem tão longas a ponto de se apresentarem como um degrau.

Como já foi ilustrado pela Figura 4(a), o trem de pulso possui dois parâmetros: a amplitude (A) e a duração (D). Para este estudo essas variáveis terão valores aleatórios dentro da faixa de operação do sistema. Um requisito para a geração da carga de trabalho é o modelo de desempenho do sistema, o qual, por meio da análise em frequência, quantifica a faixa de valores significantes. O tempo de assentamento (*settling time*) do sistema é 39 s (Eq. 1). Portanto, D deve possuir pelo menos tamanho 40 s de forma que o sistema atinja diferentes patamares de estacionariedade. Também é importante que a rajada ocorra dentro da região de operação ($U \in [40\%, 60\%]$), A deve estar entre 1.371 e 2.158 req/s.

Um sinal do tipo *Pseudo-Random Binary Sequence* (PRBS) apresenta uma série que se assemelha com o trem de pulso já apresentado, porém com duração (D) variável. O comprimento do sinal e a quantidade de mudanças de magnitude dependem do parâmetro n . No caso do PRBS esse valor corresponderá a $l = 2^n - 1$ e $m = 2^n/2$, em que l é o comprimento do sinal, m é a quantidade de variações que ocorrerá no sinal e n a quantidade de *bits* utilizados para gerar a sequência. Isso significa que, caso n seja 4 (quatro), o sinal produzirá uma entrada sintética que corresponderá a uma excitação de



(a) Sinal pseudo-aleatório com amplitude variantes dentro da região de operação.



(b) Sinal ruidoso extraído do sistema, oriundo de execução única com carga de trabalho estacionária.

Figura 6. Sinais para composição do sinal de rajada.

duração 31 s e a quantidade de vezes em que o sinal alternará será 16.

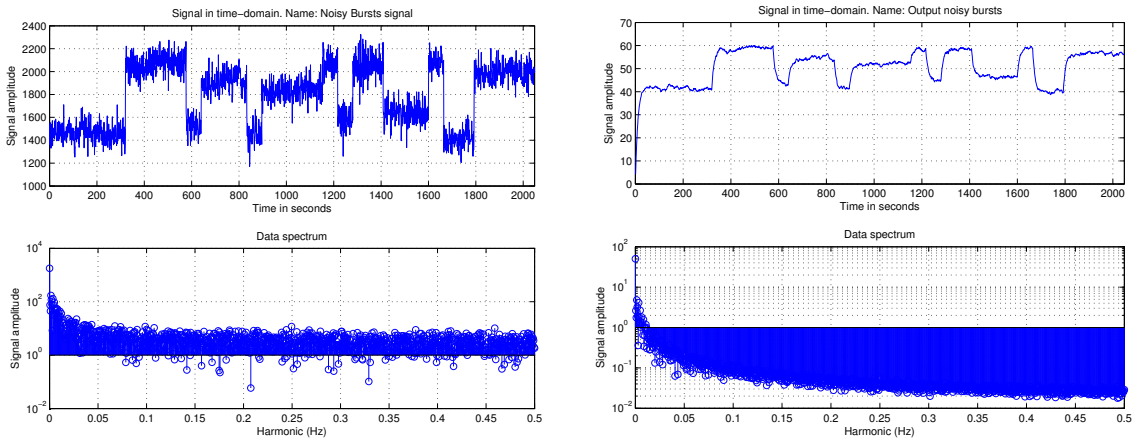
A amplitude pode ser alterada após a geração do sinal PRBS. Uma estratégia é gerar m valores dentro da faixa de operação e, para cada mudança presente no sinal, atribuir determinada amplitude. Com essa alteração o sinal resultante teria as características aleatórias de mudanças presentes no PRBS, porém não limitado aos valores extremos. A Figura 6(a) apresenta um sinal gerado dessa forma. A Figura 6(b) corresponde a uma execução de 2.048 s em regime estacionário, esse sinal é denominado como um sinal de entrada com ruído inerente da carga de trabalho.

Os espectros apresentados na Figura 6, obtidos pela FFT, podem ser somados. A Figura 7(a) corresponde ao sinal produzido. Ao passar esse sinal para a FT, o resultado produzido é apresentado na Figura 7(b). Ao inspecionar os espectros da Figura 7, percebe-se que parte considerável de frequências são removidas, comportamento típico de um filtro passa baixas. A rajada gerada conforme especificada teve seu objetivo atingido, provocando os patamares estacionários desejados. É possível mudar a duração das alterações e prever o desempenho do sistema por meio de simulação.

6. Carga de Trabalho Baseada em Traços

Como demonstrado nas seções anteriores, frequências do sinal de entrada são reduzidas após o processamento realizado pelo sistema. Portanto, ao filtrar o sinal de entrada, W_1, por um filtro com as mesmas propriedades dinâmicas do sistema, o resultado produzido, W_f, tende a ter um impacto semelhante ao de W_1. Isto é, para o sistema $F(z)$, as entradas W_1 e W_f produzem o mesmo resultado. Pode-se derivar um filtro que mantenha a mesma magnitude entre os sinais de entrada e saída (o que corresponde a uma FT de ganho unitário) e remova as frequências conforme informações obtidas a partir do Diagrama de Bode da FT identificada. O equacionamento para esse filtro é $S_{filtro}(z) = S(z)/dc$, em que dc é o ganho da FT. Considerando a FT da Eq. 1 cujo ganho DC é $0,0283$, tem-se

$$F_{filtro}(z) = \frac{F(z)}{0,0283} = \frac{0.00273}{0.02835 - 0.02562z^{-1}}. \quad (2)$$



(a) PRBS somado ao ruído da carga. (entrada)

(b) Saída correspondente.

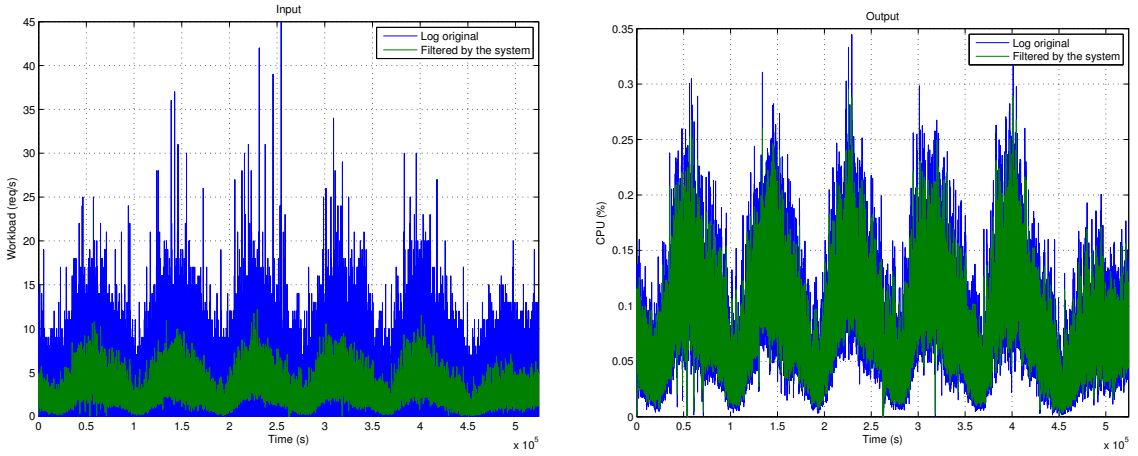
Figura 7. Rajada produzida e seu impacto no sistema.

Foram considerados 4 (quatro) traços de execução com a finalidade de verificar o impacto dessas cargas no sistema em teste, a saber: Clarknet, NASA, Copa '98 (obtidos de [LBL 2016]) e Google [Reiss et al. 2011]. Os *datasets* preparados possuem informações de requisições a servidores Web (Clarknet, NASA e Copa '98) com uma taxa de amostragem de $1\,\mathrm{s}$ ($1\,\mathrm{Hz}$) e *jobs* para Google com precisão de milissegundos ($1\,\mathrm{MHz}$). Processou-se os arquivos de modo que cada requisição presente nos arquivos representasse uma requisição a ser transformada pela FT. No caso do Google, agrupou-se as requisições em períodos de $1\,\mathrm{s}$. Dessa forma, os *datasets* serviram como sinal de entrada para duas FTs $F(z)$, Eq. 1, e $F_{filtro}(z)$, Eq. 2 — simuladas com o Matlab ©.

A Figura 8 apresenta os resultados obtidos do traço Clarknet. Ao observar os sinais de entrada (Figura 8(a)), eles são diferentes e apresentam correlação de $0,6478$. Isso significa que parte considerável do sinal original foi removida, restando apenas as frequências mais baixas. O que se observa na saída (Figura 8(b)) é que há uma semelhança grande, correlação de $0,9509$. Uma conclusão para este caso é que duas entradas distintas produzem a mesma resposta para o sistema em estudo. As diferenças nos sinais de entrada impactam de forma irrelevante no desempenho. No entanto, esse não é o caso da carga Copa '98, em que os sinais de entrada apresentam correlação de $0,9980$ e saída de $0,9998$ — o filtro removeu pouco das frequências existentes no sinal original.

A Tabela 3 apresenta os resultados obtidos de todos os traços estudados. Os traços Clarknet e NASA possuem frequências mais altas, que são absorvidas pela inércia do sistema. Já o traço da Copa '98 possui frequências baixas na alteração da taxa de req/s, isso faz com que os resultados com e sem filtro sejam equivalentes. Os traços da Google possuem altas frequências, mas também não linearidades, fazendo com que a correlação entre as entradas seja de aproximadamente $0,43$, no entanto, as saídas com aproximadamente $0,82$. Como o modelo é linear, as não linearidades existentes diminuem a correlação entre as saídas obtidas. Deve-se mencionar que essas afirmações são válidas para a FT em estudo (Eq. 1). Para sistemas com dinâmicas diferentes os resultados serão outros.

A carga de trabalho Clarknet tem pouco impacto no sistema, causando uma taxa de utilização de menos de $0,35\%$. Caso fosse necessário excitar o sistema com essa carga de trabalho, de forma tradicional, dificilmente obteríamos um resultado significante. No



(a) Entrada. (b) Saída.

Figura 8. Resultados obtidos a partir do traço Clarknet.

entanto, como há a FT do sistema e sabe-se seu comportamento em resposta a frequência, é possível observar o espectro da carga em estudo e alterar a magnitude das frequências de modo que elas impactem no sistema conforme desejado.

7. Conclusões

Este artigo apresentou uma abordagem para a modelagem de desempenho baseada em Função de Transferência (FT), bem como a análise de resposta em frequência de um sistema de *e-commerce*. O modelo utilizado possibilita a previsão do desempenho do sistema em cargas variantes no tempo e permite que cenários do tipo "*what-if*" sejam simulados através da FT sem o ônus de execuções extras. A análise de resposta em frequência permitiu quantificar como a carga de trabalho é absorvida pelo sistema, evidenciando como as frequências são atenuadas. A abordagem adotada proporcionou um novo olhar acerca da geração de carga de trabalho ao incluir o modelo do sistema no processo. Os resultados obtidos permitiram (1) a especificação de um método para a geração de rajadas que pode ser configurado de acordo com a dinâmica de desempenho do sistema e (2) a demonstração de que duas cargas distintas impactaram de forma equivalente no sistema, destacando a importância de considerar o sistema ao utilizar a carga de trabalho no processo de *benchmark*. Os métodos apresentados neste artigo podem ser generalizados para sistemas em que o modelo de desempenho seja em função da carga de trabalho.

Como trabalhos futuros, espera-se estender a modelagem para sistemas não lineares (NARX) e de múltiplas entradas e saídas (MIMO). A resposta em frequência pode ser útil na implementação de políticas de *auto scaling* para evitar a criação de VMs durante o período transiente [da Luz et al. 2016]. Outro trabalho a ser explorado é a comparação

Tabela 3. Correlações entre os sinais de entradas e saídas.

Nome	Entradas	Saídas
Clark	$0,6478$	$0,9509$
NASA	$0,5357$	$0,9198$
Copa 98	$0,9980$	$0,9998$
Google	$0,4330$	$0,7931$

de técnicas tradicionais de modelagem de carga de trabalho com a abordagem proposta neste artigo, permitindo evidenciar as vantagens e desvantagens de cada técnica.

Agradecimentos

Agradecemos CAPES, FAPESP, CNPq, LaSDPC/USP e IFSP pelo apoio financeiro, e também aos revisores anônimos por suas contribuições.

Referências

Centurion, A. M., Santana, M. J., Santana, R. C., and Bruschi, S. M. (2012). Impacto da carga de trabalho com rajadas no desempenho de serviços web. In *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 422–435.

da Luz, H. J. F., Júnior, L. A. P., dos Santos de Souza, F. L., and Monaco, F. J. (2016). Modelagem analítica de sobrecarga transiente em sistemas computacionais por meio de parâmetros dinâmicos obtidos empiricamente. In *XIV Workshop de Computação em Clouds e Aplicações*, Salvador, BA. Sociedade Brasileira de Computação.

Diao, Y., Hellerstein, J. L., Parekh, S., Shaikh, H., and Surendra, M. (2006). Controlling quality of service in multi-tier web applications. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, pages 25–25.

Hellerstein, J. L., Diao, Y., Parekh, S., and Tilbury, D. M. (2004). *Feedback Control of Computing Systems*. Wiley-IEEE Press.

Huang, D., He, B., and Miao, C. (2014). A survey of resource management in multi-tier web applications. *IEEE Communications Surveys Tutorials*, 16(3):1574–1590.

Jain, R. (1990). *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons.

Jiang, Y., Sivalingam, L. R., Nath, S., and Govindan, R. (2016). Webperf: Evaluating what-if scenarios for cloud-hosted web applications. In *Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference*, New York, NY, USA. ACM.

LBL (2016). The internet traffic archive. `http://ita.ee.lbl.gov/`. Accessado: 01/12/2016.

Ljung, L. (1999). *System Identification: Theory for the User*. Pearson Education.

Mamani, E. L. C. (2016). *Metodologia de benchmark para avaliação de desempenho não-estacionária: um estudo de caso baseado em aplicações de computação em nuvem*. PhD thesis, PPG-CCMC ICMC USP.

Mamani, E. L. C., Pereira, L. A., Santana, M. J., Santana, R. H. C., Nobile, P. N., and Monaco, F. J. (2015). Transient performance evaluation of cloud computing applications and dynamic resource control in large-scale distributed systems. In *High Performance Computing Simulation (HPCS), 2015 International Conference on*, pages 246–253.

Mell, P. M. and Grance, T. (2011). Sp 800-145. the nist definition of cloud computing. Technical report, National Institute of Standards & Technology, Gaithersburg, USA.

Menasce, D. (2002). TPC-W: a benchmark for e-commerce. *IEEE Internet Computing*.

Papadopoulos, A. V., Maggio, M., Terraneo, F., and Leva, A. (2015). A dynamic modelling framework for control-based computing system design. *Mathematical and Computer Modelling of Dynamical Systems*, 21(3):251–271.

Pereira, L. A. (2016). *Uma abordagem baseada em resposta em frequência para modelagem e avaliação de desempenho não estacionária em sistemas computacionais*. PhD thesis, PPG-CCMC ICMC-USP.

Pereira, L. A., Mamani, E. L. C., Santana, M. J., Santana, R. H. C., Nobile, P. N., and Monaco, F. J. (2015a). Non-stationary simulation of computer systems and dynamic performance evaluation: A concern-based approach and case study on cloud computing. In *Computer Architecture and High Performance Computing (SBAC-PAD), 2015 27th International Symposium on*, pages 130–137.

Pereira, Jr., L. A., Mamani, E. L. C., Santana, M. J., Santana, R. H. C., Monaco, F. J., and Nobile, P. N. (2015b). Extending discrete-event simulation frameworks for non-stationary performance evaluation: Requirements and case study. In *Proceedings of the 2015 Winter Simulation Conference*, WSC '15, Piscataway, NJ, USA. IEEE Press.

Qu, C., Calheiros, R. N., and Buyya, R. (2016). Auto-scaling web applications in clouds: A taxonomy and survey. *arXiv Computing Research Repository (CoRR)*.

Reiss, C., Wilkes, J., and Hellerstein, J. L. (2011). Google cluster-usage traces: format + schema. Technical report, Google, Inc. Posted at `https://github.com/google/cluster-data`.

Santos, B., Carnivali, G., Almeida, W., Vieira, A. B., Ítalo Cunha, and Almeida, J. (2016). Caracterização do comportamento dos clientes de um sistema de vídeo ao vivo durante um evento de larga escala na internet. In *XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 468–481.

Silva, T. H., de Melo, P. O. S. V., , Almeida, J., and Oliveira, A. A. F. (2013). Uma fotografia do instagram: caracterização e aplicação. In *XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 455–468.

Souza, F. (2016). Extensão da geração de carga do Bench4Q para benchmark de desempenho em regime transiente. Master's thesis, ICMC/USP.

Veeraraghavan, K., Meza, J., Chou, D., Kim, W., Margulis, S., Michelson, S., Nishtala, R., Obenshain, D., Perelman, D., and Song, Y. J. (2016). Kraken: Leveraging live traffic tests to identify and resolve resource utilization bottlenecks in large scale web services. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 635–651, GA. USENIX Association.

Yang, F. and Liu, J. (2012). Simulation-based transfer function modeling for transient analysis of general queueing systems. *European Journal of Operational Research*.

Yuan, D., Joshi, N., Jacobson, D., and Oberai, P. (2013). Scryer: Netflix's predictive auto scaling engine - part 2. `https://goo.gl/6tlHdb`. Accessado: 01/12/2016.

Zhang, W., Wang, S., Wang, W., and Zhong, H. (2011). Bench4q: A qos-oriented e-commerce benchmark. In *Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual*, pages 38–47.

Trilha Principal do SBRC 2017
Sessão Técnica 22
Redes Centradas em Conteúdo

DIRESC: Um protocolo para descoberta e recuperação de dados em redes centradas em conteúdo e tolerantes a atraso

Cláudio Diego Souza[1], **Danielle L. Ferreira**[1], **Carlos A. V. Campos**[1]

[1]Programa de Pós-Graduação em Informática (PPGI)
Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
Rio de Janeiro – RJ – Brasil

{diego.souza, danielle.ferreira,beto}@uniriotec.br

Abstract. *Content-centric networks (CCN) has emerged as an innovative paradigm that modifies the traditional host-centric and address-based communication model. Its application to DTN scenarios is known to positively impact some network metrics and characteristics, such as delay and intermittence. Thus, in this paper we justify the feasibility of this combination and present DIRESC, a protocol for selection of message relays for data announcement, discovery and retrieval through a scheme of social cooperation between nodes. We implement our proposal using the ONE simulator and compared its performance against CCN broadcasting and PIFP protocols. In our results, DIRESC presented a decrease of at least* 73.3% *in data recovery delay and a gain of at least* 10.1% *in satisfaction of content requests over PIFP. Thus, we present DIRESC as a promising solution for accelerating the process of obtaining data in this type of network.*

Resumo. *Redes centrada em conteúdo (CCN) emergiram como um paradigma inovador que modifica o modelo tradicional de comunicação centrada no host e endereçamento. Sua aplicação aos cenários de DTN é conhecida por ter impacto positivo em algumas métricas e características de rede, como atraso e intermitência. Assim, neste artigo*[1] *justificamos a viabilidade desta combinação e apresentamos o DIRESC, um protocolo para a seleção de retransmissores de mensagem para o anúncio, descoberta e recuperação de dados através de um esquema de cooperação social entre nós. Implementamos nossa proposta usando o simulador ONE e comparamos seu desempenho com os protocolos CCN broadcasting e PIFP. Em nossos resultados, DIRESC apresentou uma redução de pelo menos US* $73,3\%$ *no atraso na recuperação de dados e um ganho de pelo menos US* $10,1\%$ *na satisfação dos pedidos de conteúdo sobre o PIFP. Assim, apresentamos o DIRESC como uma solução promissora para acelerar o processo de obtenção de dados neste tipo de rede.*

1. Introdução

Redes tolerantes a atraso ou interrupções (*Delay/Disruption Tolerant Networking* - DTN) preveem a existência de cenários de rede sem fio sem infraestrutura e de múltiplos saltos nos quais não se garante a existência de caminhos fim-a-fim entre o par origem-destino. Trata-se de ambientes de rede auto-organizáveis, nos quais tais dispositivos se comunicam

[1]Os dois primeiros autores deste trabalho recebem bolsa de estudos da CAPES.

de maneira oportunista, ou seja, valendo-se de oportunidades de contato entre si para encaminhar pacotes ao longo da rede [Conti et al. 2015]. Neste ambiente, todos os nós são dispositivos finais, como celulares, tablets, notebooks etc, no entanto, se comportam como roteadores armazenando os pacotes recebidos e aguardando a oportunidade de retransmitílos a outros nós.

Em razão das características destas redes, como a alta dinamicidade das topologias, intermitência e baixa conectividade [Conti et al. 2015], os protocolos de roteamento tradicionais da arquitetura TCP/IP não se aplicam neste contexto, isto é, não é funcional a implementação de protocolos que preveem comunicações fim-a-fim em cenários nos quais ela é inexistente. Portanto, soluções de rede alternativas para tais ambientes têm tentado fugir de abordagem centradas em endereçamento para abordagens centradas em dados [Amadeo et al. 2014].

Uma solução possível é a utilização da arquitetura de redes centradas em conteúdo (*Content-Centric Networking - CCN*), conceitualmente introduzida por [Jacobson et al. 2009], que tem emergido como uma promissora abordagem de rede para a recuperação de dados a partir da desconstrução do conceito de endereçamento no processo de busca por provedores de serviços. Diferentemente das redes centradas em hospedeiro (*host-centric*), CCN foca na identificação do conteúdo buscado, modificando o paradigma de comunicação para uma rede centrada em dados (*data-centric*).

Em CCN, a comunicação é conduzida pelos nós consumidores ou requisitantes que, quando interessados em determinado conteúdo, disseminam pacotes de interesse na rede. Quando estes pacotes atingem um nó que possua o conteúdo buscado, tal nó envia os pacotes de dados ao requisitante, concluindo o processo de busca e entrega. Além do encaminhamento de interesses baseado em nomes de conteúdos, o paradigma CCN se caracteriza também pela replicação de pacotes de dados em cache na rede, o que facilita a entrega destes pacotes ao requisitante [Jacobson et al. 2009].

Os nós que compõem um cenário CCN possuem três componentes básicos [Jacobson et al. 2009]: a *content store* (CS), que é a cache para armazenamento de dados; a *forwarding information base* (FIB), uma tabela que mapeia os anúncios de provedores de conteúdo à interface pela qual tal anúncio foi recebido; e a *pending interest table* (PIT), uma tabela que registra as requisições de conteúdos realizadas e ainda não satisfeitas, mapeando-as às interfaces por onde tais requisições chegaram.

Em [Liu et al. 2016] são destacados alguns benefícios que o paradigma CCN é capaz de prover ao paradigma DTN. Dentre eles, destaca-se o fato de CCN permitir comunicações assíncronas entre um par requisitante-provedor, isto é, sem a necessidade de que estes nós estabeleçam e mantenham uma comunicação fim-a-fim. Além deste, a busca de dados por nomes facilita a sua recuperação em ambientes de alta mobilidade, quando não se sabe a localização do nó, como ocorre em DTN. Por fim, a replicação de dados na rede reduz o número de requisições e o atraso na recuperação destes dados.

Apesar dos benefícios que a aplicação CCN é capaz de prover no contexto de DTN, a sua combinação levanta questões de pesquisa que ainda precisam ser estudadas. Em DTN, um desafio chave de roteamento é determinar a estratégia apropriada de seleção do nó retransmissor a fim de aumentar a probabilidade de entrega de pacotes, minimizando o número de replicações na rede e reduzindo ao máximo o atraso desta en-

trega. O protocolo Epidêmico [Becker et al. 2000], que é baseado no encaminhamento de pacotes em toda oportunidade de contato, apresenta bons resultados em termos de atraso e taxa de entrega, mas acarreta um alto custo em replicações de pacotes e, portanto, sobrecarga na rede [Zhu et al. 2015]. Já as comunicações em CCN foram idealizadas para operar em *broadcast* [Jacobson et al. 2009]. Uma tendência nos trabalhos mais recentes que utilizam DTN e CCN conjuntamente é encontrar um meio termo entre a política de funcionamento de um (neste caso, CCN, conceitualmente projetado para operar em *broadcast*), às limitações de outro (DTN, que precisa evitar sobrecarga e exaustão de recursos nos nós), e assim tornar os processos de encaminhamento mais "conscientes" por meio de estratégias como a seleção de retransmissores, a fim de evitar o comportamento epidêmico na comunicação entre os nós.

A estratégia de seleção de nós retransmissores para a descoberta e recuperação de conteúdo se propõe a limitar o número de nós que recebem pacotes por meio de uma escolha baseada em algum critério. Com base nesta estratégia, no presente artigo propomos o DIRESC, um protocolo para anúncio, descoberta e recuperação de dados em redes centradas em conteúdo e tolerantes a atraso (*Content-Centric Delay Tolerant Network* - CCDTN) baseado na cooperação entre nós que possuem determinado grau de relacionamento social.

Nossa proposta se baseia na seleção de nós retransmissores, uma vez que, por meio dela, evita-se o envio de mensagens a nós de modo indiscriminado na rede, o que apresenta um impacto significativo na quantidade de pacotes replicados e transmissões realizadas. Assim, recursos como energia e capacidade de *buffer* podem ser poupados. O esquema de cooperação social proposto se baseia no relacionamento existente entre os nós, calculado por meio do seu histórico de encontros. Este relacionamento é capaz de influenciar a probabilidade de entrega de mensagens, em vista que permite inferir se um par de nós específico possui chances de se reencontrar na rede, qualificando-os ou não como nós aptos à retransmissão, o que influencia também no atraso de recuperação de conteúdo. Nossa proposta define três possibilidades de envio de requisições e outras três para entrega de conteúdo, diferentemente de outras abordagens que, além de dar ênfase no processo de descoberta (envio de interesses), dando menor destaque ao processo de entrega de dados, definem um único critério para determinar a competência de um nó em entregar um pacote. Assim, utilizamos a estratégia de seleção de retransmissores de maneira mais distribuída e, portanto, sem exigir intensamente de apenas um conjunto de nós específico na rede.

Assim, nossa proposta se divide em três etapas principais para o funcionamento da rede: anúncio, descoberta e recuperação de conteúdo. Cada uma delas leva em consideração o fator de relacionamento social entre nós para o seu funcionamento e, de modo geral, visam reduzir a distância (temporal e em número de saltos) entre um requisitante e o conteúdo solicitado.

As principais contribuições deste trabalho podem ser resumidas da seguinte forma:

1. Proposta do protocolo DIRESC para anúncio, descoberta e recuperação de conteúdo em CCDTN. O protocolo proposto obteve um aumento no percentual de satisfação de requisições, ou seja, na relação entre interesses enviados e atendidos, mostrando resultados melhores quando comparado com outras propostas. Além disso, o DI-

RESC apresentou um alto decaimento no atraso médio da recuperação de dados quando comparamos nossa proposta com outras abordagens.

2. A implementação da arquitetura CCN e do protocolo DIRESC no ambiente de simulação ONE (*Opportunistic Network Environment*), o qual foi projetado para representar redes toletantes a atraso.

O restante do artigo está estruturado da seguinte forma. A Seção 2 aborda trabalhos anteriores relacionados a nossa proposta. A Seção 3 descreve o funcionamento do protocolo DIRESC. A Seção 4 apresenta a avaliação de desempenho realizada sobre o protocolo proposto e uma comparação com outros trabalhos da literatura. Por fim, a Seção 5 apresenta as conclusões e trabalhos futuros.

2. Trabalhos relacionados

No que concerne a combinação de CCN com redes sem fio sem infraestrutura e de múltiplos saltos, [Kim and Ko 2015] elabora um mecanismo proativo de atualização periódica de informações sobre nós da rede, e outro que propõe a transmissão de mensagens para recuperação confiável de conteúdo baseada na classificação destes nós, além de fazer uso da política de seleção de nós retransmissores, que neste caso se baseia na distância em número de saltos até o destino e qualidade do enlace. A proposta atinge 50% de ganho em taxa de trasferência quando comparada com o método básico de encaminhamento oportunista por *broadcast*.

Por sua vez, em [Anastasiades et al. 2014] é apresentada uma proposta baseada em agente, na qual um nó requisitante, por meio de um processo de sondagem e comunicação *handshake* de três vias, encontra nós agentes em potencial (de acordo com informações como histórico de locais visitados) e delega a um deles a função de descoberta e recuperação de conteúdo em uma rede CCN oportunista. Os resultados do trabalho apontam que um intervalo de sondagem de 30s apresenta o melhor desempenho no cenário proposto em vista que o número de notificações pode ser diminuído em até 80%.

O trabalho de [Lu et al. 2014] propõe o STCR (*Social-Tie based Content Retrieval*) que, por meio do cálculo de relacionamento e centralidade social e do algoritmo de *clustering K-means*, agrupa em *clusters* nós de centralidade social semelhante. Neste contexto, anúncios de provedores de conteúdo e requisições convergem para os nós de maior centralidade social. Já o envio destas requisições (de um nó de mais alta centralidade até um provedor) e os de dados (do provedor ao requisitante) é realizado através de nós que possuam alto relacionamento social com o destinatário do pacote em questão. A proposta apresenta ganhos em termos de atraso, custo e taxa de entrega em relação aos protocolos *Epidemic, BubbleRap e PeopleRank*. Por outro lado, o STCR, por seccionar o processo de busca por dados em duas etapas (enviar intererres primeiramente para os nós de mais alta centralidade e só então enviá-los para os provedores de conteúdo), aumenta o tempo de recuperação de dados em comparação com outros mecanismos para CCN aplicada no cenário DTN, como é verificado em [Duarte et al. 2015].

Baseado no protocolo PROPHET, de roteamento em DTN, [Duarte et al. 2015] elabora o PIFP (*Probabilistic Interest Forwarding Protocol*), um protocolo probabilístico de encaminhamento de pacotes de interesse em DTN, a partir do qual os nós calculam a probabilidade de encontrarem determinado conteúdo e utilizam esta informação para selecionar retransmissores de interesses. Isto é, nós requisitantes de conteúdo apenas

enviam interesses para outros nós que possuam uma probabilidade superior a sua. Já o envio de pacotes de dados possui duas variantes: no PIFP, este envio considera apenas o caminho reverso ao envio dos interesses; já no PIFP *Proative*, os dados são enviados a qualquer nó da rede. O trabalho apresenta ganhos em termos de número de interesses enviados e atendidos e atraso de resposta em relação ao STCR. No entanto, por convocar sempre os nós com maiores probabilidades, o PIFP negligencia a competência de outros nós, oferecendo uma seleção não balanceada de retransmissores.

Em uma análise mais acurada dos trabalhos supracitados, podemos levantar uma questão: suas abordagens não estariam limitando tão fortemente as replicações de interesses a ponto de prejudicar a rede em termos de atraso de entrega e percentual de satisfação de requisições? Além disso, dado o ranqueamento dos "melhores"nós da rede segundo algum critério, seja centralidade social ou probabilidade de encontro com conteúdo, a constante seleção dos nós mais bem posicionados no ranqueamento produz uma "distribuição injusta de carga", ou seja, faz com que a maior parte do tráfego da rede seja realizada apenas por uma pequena parcela de nós, sobrecarregando-os e exaurindo rapidamente seus recursos, como explica [Junior et al. 2014].

3. O protocolo DIRESC

Visando superar os desafios da combinação de DTN e CCN e a questão levantada na seção anterior, propõe-se o DIRESC, um protocolo para descoberta e recuperação de conteúdo baseado em cooperação social, que tem por objetivo principal reduzir o atraso de descoberta e recuperação de conteúdo em redes CCDTN. Diferentemente de outras abordagens propostas na literatura, nas quais os conteúdos são alcançados através da convergência de pacotes de interesse por meio do que [Junior et al. 2014] chama de "escolha de nós preferidos", uma vez que definem um único critério para determinar a competência de um nó em entregar um pacote, o DIRESC propõe uma abordagem distribuída, evitando assim tal comportamento indesejável, resultante do funcionamento de protocolos oportunistas com base social. A definição de três possibilidades de envio de requisições e outras três para entrega de dados evita o comportamento em *broadcast*, típico de CCN, mas torna a proposta mais tolerante em termos de critérios de seleção de nós retransmissores.

3.1. Relacionamento social

O fator de relacionamento social $R_i(j)$, entre o par de nós i e j, é uma medida calculada de acordo com o histórico de encontros entre os mesmos. Para a definição do seu valor, leva-se em consideração o quão recente os encontros ocorreram e a sua quantidade, ou seja, a frequência com a qual estes nós entram em contato. O valor de $R_i(j)$, baseado em [Lu et al. 2014], é definido da seguinte forma.

$$R_i(j) = \sum_{k=1}^{N} F(t_{base} - t_{jk}) \tag{1}$$

A influência de cada encontro entre os nós i e j, sendo N o total de encontros entre eles até o instante t_{base}, é determinado por ***F(x)***, uma função que calcula o impacto gerado em $R_i(j)$ pelo quão recente estes encontros ocorreram, em que x é o intervalo de tempo entre um encontro no passado e o tempo atual. A função $F(x)$ é definida da seguinte maneira:

$$F(x) = (\frac{1}{2})^{\lambda x}, \text{onde } \lambda = e^{-4}. \tag{2}$$

O protocolo DIRESC utiliza o cálculo do relacionamento social $R_i(j)$ como princípio. No entanto, diferentemente de [Lu et al. 2014], que utiliza essa medida para a formação de agrupamentos entre nós que possuem valor de centralidade social similar, o DIRESC utiliza o valor de $R_i(j)$ como um parâmetro para determinar se um par de nós possui um alto relacionamento social entre si. Utilizamos uma abordagem diferenciada do relacionamento social uma vez que, por se valer da métrica de centralidade, [Lu et al. 2014], além de ocasionar o problema da escolha de "nós preferidos", tem seu processo de descoberta de conteúdo seccionado, elevando o atraso de recuperação dos dados.

Portanto, o DIRESC compara $R_i(j)$ com a sua média, $R_{medio} = \frac{1}{N} \sum_k R_i(k)$, no qual $k = 1, 2, ..., N$ e N é o número total de nós com os quais i teve contato até o instante da conexão com j. Logo, se $R_i(j) \geq R_{medio}$ então i e j possuem alto relacionamento social.

A fim de manter o fator de relacionamento social sempre atualizado, a todo contato entre dois nós, os mesmos atualizam o valor de $R_i(j)$ e, identificando que esses nós possuam um alto relacionamento social eles passam a ser considerados "amigos"e a tomar decisões que definirão o comportamento da rede. Para a melhor compreensão destas decisões, elas foram enquadradas em três fases, definidas como: (i) distribuição, (ii) busca e (iii) retorno.

3.2. Fase de distribuição

A fase de distribuição compreende a disseminação de informações acerca de conteúdos. Nela ocorrem dois tipos de anúncios: no primeiro, nós produtores, ou seja, nós que geram pacotes de dados, informam para nós com quem têm alto relacionamento social quais são os dados que possuem em CS; no segundo, nós que receberam anúncios de produtores comunicam a seus amigos tal recepção. Assim, delimitam-se agrupamentos capazes de responder às solicitações para tais conteúdos. Portanto, nesta etapa, assumindo que dois nós sejam amigos, segue-se as seguintes trocas de anúncios:

1. Anúncio direto: caso o nó i seja um produtor e possua conteúdos na CS não registrados na FIB de j, ou não esteja mapeado como um nó de próximo salto caso tais registros de fato existam, o nó j atualiza sua FIB, criando registros que façam o mapeamento entre o nó i e os prefixos que ele atende, ou inserindo tal nó nos registros existentes como um próximo salto para envio de interesses. Ou seja, o nó j passa a compreender que i é capaz de atender solicitações para determinada porção de conteúdos.

2. Anúncio transitivo: caso a FIB do nó i possua registros que não existem na FIB de j, esta é atualizada com os registros faltantes de modo que i seja mapeado como um nó de próximo salto para o envio de interesses.

Deste modo, uma vez que existe alto relacionamento social entre nós, criam-se agrupamentos capazes de responder à solicitações de forma ágil, tendo em vista do elevado nível de contatos entre os nós que os compõem. O intuito de um agrupamento é, portanto, reduzir o tempo de descoberta de conteúdos a partir do momento em que os pacotes de interesse o atingem. A Figura 1 sumariza as tarefas de anúncios no DIRESC.

3.3. Fase de busca

A fase de busca, por sua vez, diz respeito a requisição de conteúdo. Nela, selecionamos nós que se enquadrem em três critérios para envio de interesses, a fim de evitar que estes

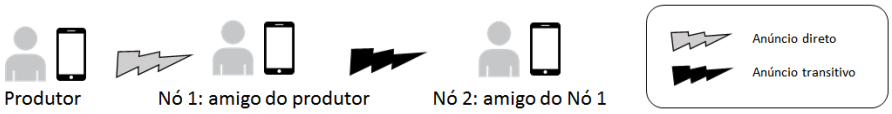


Figura 1. Anúncio direto e transitivo

sejam transmitidos de forma epidêmica, mas dando certa tolerância para que tais envios ocorram. Leva-se em consideração informações que os nós precisam registrar acerca de requisições. Primeiramente, define-se dois contadores de retransmissões para cada requisição e um valor máximo que cada um pode atingir. Cada contador se designa a um grupo específico para o qual se pode enviar pacotes de interesse. São eles: o agrupamento de nós que possui informações em FIB sobre certo conteúdo; e nós de alto relacionamento social com o detentor do pacote de interesse. Assim, um nó, de posse de um interesse, somente pode enviá-lo até que o número de envios por grupo atinja o número máximo permitido, evitanto que se eleve o número de replicações deste pacote exacerbadamente. Desta forma, os nós transmitem pacotes de interesse caso quaisquer das premissas a seguir seja verdadeira, como observado na Figura 2:

1. O nó encontrado possui os dados armazenados na CS. Neste caso, não se contabiliza o envio de interesse (ver Figura 2(a)).

2. O nó encontrado pertence ao agrupamento de nós que possui informações na FIB a respeito de tal conteúdo e o número de envios da requisição em questão para este grupo é menor que o valor máximo permitido. Neste caso, registra-se na PIT uma nova retransmissão da requisição pelo critério de agrupamento (ver Figura 2(b)).

3. Os nós possuem um alto relacionamento social entre si e o número de envios da requisição entre eles é menor que o valor máximo permitido. Neste caso, registra-se na PIT uma nova retransmissão da requisição pelo critério de amizade. Este critério de envio aumenta a persistência de pacotes de interesse em descobrir conteúdo na rede, de modo que os nós que se encaixam nesta avaliação trabalham de modo socialmente cooperativo buscando dados em favor do solicitante, tendo em vista que este par de nós possui grandes chances de se reencontrar (ver Figura 2(c)).

Além disso, dados os critérios supracitados para envio de pacotes de interesse, utiliza-se um esquema de priorização de envio, no qual um nó interessado em determinado conteúdo dá prioridade de envio de solicitações a nós que se enquadram nos critérios acima, na ordem em que eles se apresentam.

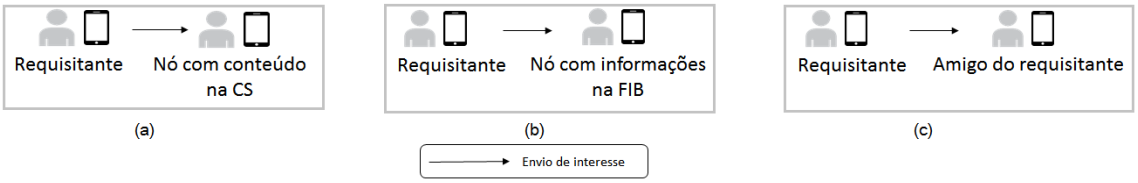


Figura 2. Possibilidades de envio de pacotes de interesse

3.4. Fase de retorno

Assim como na fase de busca, na fase de retorno, selecionamos nós que se enquadrem em três critérios para envio de dados, evitando que estes sejam transmitidos de forma

epidêmica, e dando tolerância para o envio destes pacotes, aumentando suas chances de entrega. Esta fase se refere a entrega de conteúdo, ou seja, quando um nó que possui na CS o pacote de dados correspondente a uma solicitação recebida, ele enviará tais dados a outros nós baseado nos seguintes critérios, como observado na Figura 3:

1. Para todo nó que lhe fez tal solicitação, ou seja, todo nó que lhe enviou um pacote de interesse para os dados que ele possui (ver Figura 3(a)).

2. Para todo nó que possua na PIT um registro para aqueles dados e, portanto, é um solicitante, ainda que não tenha lhe enviado um pacote de interesse, mas está aguardando receber os dados por meio de outro nó (ver Figura 3(b)).

3. Para todo nó que possua um alto relacionamento social com o nó requisitante original dos dados (o nó que inseriu na rede o pacote de interesse recebido por este que possui os dados). Deste modo, mais uma vez, utiliza-se o valor obtido do cálculo de relacionamento social, no entanto, para auxiliar a recuperação dos dados (ver Figura 3(c)).

Da mesma forma que na fase de busca, utiliza-se um esquema de priorização de envio de acordo com a ordenação apresentada dos critérios para entrega de dados.

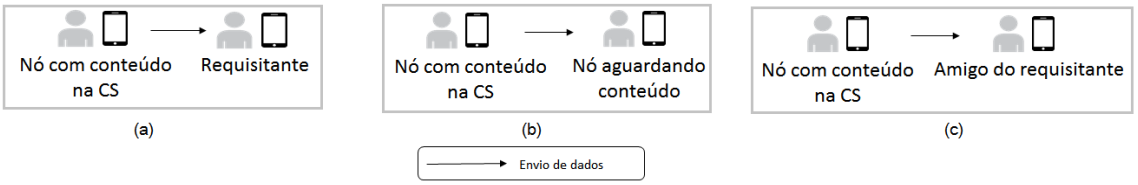


Figura 3. Possibilidades de envio de pacotes de dados

3.5. O algoritmo

O Algoritmo 1 descreve o funcionamento do DIRESC, sumarizando as tomadas de decisão dos processos de anúncio, descoberta e recuperação de dados. Cada um destes processos, identificados nas iterações do algoritmo, respectivamente, é representado por uma das fases descritas anteriormente (distribuição, busca e retorno).

4. Avaliação de desempenho

Nesta seção serão apresentados os resultados obtidos para o protocolo DIRESC, que foi experimentalmente avaliado por meio do simulador ONE (*Opportunistic Network Environment*) [Keränen et al. 2009], no qual foi implementada a arquitetura de redes CCN, de modo que se tornasse possível combinar este modelo de comunicação com a proposta do simulador, que é voltada para redes tolerantes a atraso. Para a geração dos gráficos foi utilizado o *software* R [R Core Team 2016].

O objetivo de avaliar o desempenho do DIRESC é para verificar a sua capacidade de alcançar resultados satisfatórios em termos de taxa de recuperação de conteúdo e controle da sobrecarga de pacotes que são inseridos na rede, de modo que o atraso para a obtenção dos dados seja reduzido em comparação com as outras abordagens. Em seguida, o cenário de simulação será apresentado e os resultados experimentais serão mostrados e discutidos.

Algoritmo 1: DIRESC: PSEUDOCÓDIGO

```
   Entrada: i, j, R_medio, maxEnvios
1  início
2      para todo contato de i e j faça
3          calcula-se R_i(j) (ver Equação (1))
4          % Fase de distribuição do DIRESC
5          se R_i(j) ≥ R_medio então
6              se i é produtor então
7                  para cada prefixo p na CS de i que não existe na FIB de j faça
8                      Atualize a FIB de j: adicionando o prefixo p e mapeando-o para i
9                  fim
10             fim
11             para cada registro r na FIB de i que não existe na FIB de j faça
12                 Atualize a FIB de j: adicionar uma cópia deste registo r mapeando-o para i
13             fim
14         fim
15         % Fase de busca do DIRESC
16         para cada entrada na PIT de i interessada no conteúdo c faça
17             se c está presente na CS de j então
18                 i envia um pacote de interesse em c para j
19             fim
20             senão
21                 se (há um registro na fib de j para c ∨ j é um nó de próximo salto para c na FIB de i) ∧ o
                     número de submissões desta requisição para o cluster < maxEnvios então
22                     i envia um pacote de interesse em c para j
23                     O número de submissões desta requisição para o cluster é aumentado
24                 fim
25                 senão
26                     se R_i(j) ≥ R_avg ∧ o número de envios desta requisição para amigos < maxEnvios
                         então
27                         i envia um pacote de interesse em c para j
28                         O número de submissões desta requisição para amigos é aumentado
29                     fim
30                 fim
31             fim
32         fim
33         % Fase de retorno do DIRESC
34         para cada interesse no conteúdo c recebido faça
35             se c está presente na CS de i então
36                 se j é o requisitante ∨ há uma entrada na PIT de j interessada em c ∨ Rj(requisitante) ≥
                     R_avgofj então
37                     i envia o pacote de dados c para j
38                 fim
39             fim
40         fim
41     fim
42 fim
```

4.1. Cenário de simulação

O cenário utilizado foi o mesmo descrito em [Duarte et al. 2015] e foi adotado por ser bastante diverso, com diferentes perfis de mobilidade, e para que pudéssemos comparar o DIRESC com o PIPF e CCN *broadcasting* de forma justa.

O cenário contempla a existência de 100 nós, dos quais 62 são pedestres, 32 são carros e 6 são bondes, se movimentando no mapa da cidade de *Helsinki*, Finlândia. Os nós se movem de acordo com diferentes modelos de mobilidade determinado para cada grupo de nós. Pedestres e carros se movem segundo o modelo *Shortest Path Map-Based Movement*, que utiliza o algoritmo Dijkstra para encontrar caminhos mais curtos entre dois pontos no mapa; enquanto que bondes usam o modelo *Map-Based Movement*, o qual é parametrizado por rotas específicas para determinar sua movimentação. A tecnologia utilizada foi o Bluetooth, com taxa de transmissão de 250kbps e alcance de 10m. A Tabela

1 apresenta mais detalhes acerca do cenário de simulação.

A criação das mensagens de interesse são agendadas durante a simulação por meio da classe de leitura de eventos *StandardEventsReader*, disponível no simulador ONE. Assim, através da leitura do *trace* que registra esses agendamentos, os pacotes de interesse são gerados no *buffer* de nós escolhidos aleatoriamente, em diferentes instantes de tempo, durante a simulação. A criação dos pacotes de interesse está sujeita a capacidade do *buffer* dos nós. Além disso, os nós enviam os pacotes de interesse gerados de acordo com os critérios estabelecidos pelos protocolos propostos. O DIRESC utiliza o procedimento descrito na Seção 3.3. O protocolo PIFP calcula a probabilidade que os nós possuem de encontrar determinado conteúdo e o envio de interesses é realizado para os aqueles que possuem o maior valor neste cálculo. Por sua vez, o protocolo CCN *broadcasting* se refere ao comportamento padrão da arquitetura CCN, isto é, por meio do envio de pacotes em toda oportunidade de contato.

De forma análoga, a geração dos pacotes de dados segue um procedimento similar ao descrito para as mensagens de interesse. Ou seja, os nomes de conteúdos são lidos a partir de um *trace* e os pacotes de dados correspondentes são criados na CS dos produtores. No total, são criados 500 conteúdos diferentes em 15 nós produtores igualmente definido em [Duarte et al. 2015].

Tabela 1. Parâmetros da simulação

Parâmetro	Valor
Duração	86400s (24h)
Área (metros x metros)	4500m x 3500m
Velocidade média (m/s)	Pedestres: 0.5 a 1.5, Carros: 2.7 a 13.9, Bondes: 10 a 30
Tempo de vida das requisições	8 horas
Capacidade de *buffer*	Consumidores: 50MB, Produtores: 100MB

Os resultados reportados refletem 30 rodadas de simulação para cada um dos protocolos (DIRESC, PIFP e CCN *broadcasting*). Para o cálculo das métricas apresentadas, considerou-se um intervalo de 15 minutos para contabilizar informações sobre trocas de mensagens, no qual cada observação assumiu-se o valor acumulado destas informações desde o intervalo anterior até o atual. O mesmo cenário foi utilizado para as simulações dos três protocolos supracitados.

4.2. Resultados experimentais

O protocolo DIRESC foi avaliado em termos do atraso médio e da taxa de recuperação de conteúdo, bem como do controle de sobrecarga de pacotes que trafegam pela rede. Os resultados mostram que o protocolo proposto é capaz de reduzir significativamente, cerca de 70%, o atraso médio necessário para a obtenção do conteúdo em relação a outras abordagens, com um menor tráfego de mensagens.

Com o objetivo de avaliar o protocolo DIRESC e apresentar os resultados, foram definidas as seguintes métricas:

Relação entre interesses enviados e satisfeitos: refere-se à relação absoluta e percentual entre a quantidade de conteúdo entregue aos requisitantes e o número de requisições feitas por estes. Deste modo, pode-se analisar o quão satisfatória uma proposta é em relação à quantidade de solicitações de conteúdos feitas na rede.

Carga total: definida como quantidade de pacotes de interesse e dados inseridos na rede, na qual se observa a quantidade de replicações de pacotes feita na rede.

Atraso de recuperação de conteúdo: refere-se ao intervalo de tempo médio calculado para todas as recuperações de dados, desde o momento do envio da solicitação até a entrega de conteúdo. Assim, avalia-se o quanto uma proposta é capaz de reduzir o tempo de descoberta e recuperação de dados na rede.

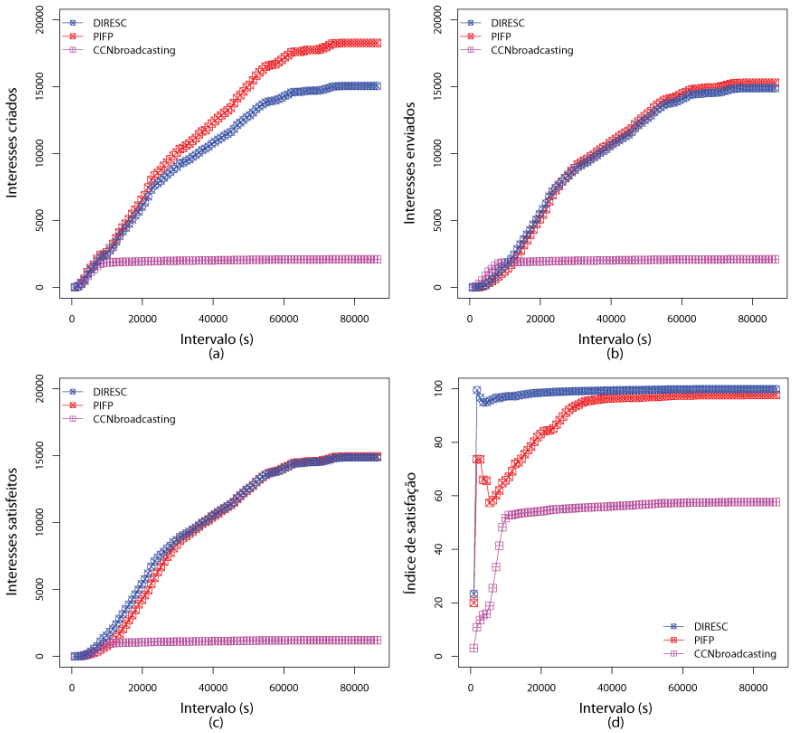


Figura 4. Relação entre interesses enviados e satisfeitos

A Figura 4 apresenta a relação entre os interesses criados, enviados e satisfeitos na rede. Considera-se uma mensagem de interesse satisfeita quando o nó requisitante que gerou uma mensagem de interesse recebe o conteúdo requisitado. A quantidade de mensagens de interesse criadas na rede é mostrada na Figura 4(a). Nela, observa-se que esta quantidade é semelhante para os protocolos DIRESC e PIFP, diferentemente do protocolo CCN *broadcasting* que exibe uma quantidade bem reduzida das mensagens de interesse criadas. Esse comportamento pode ser explicado pela característica do protocolo CCN *broadcasting*, na qual a disseminação das mensagens de interesse é feita a todo contato entre nós. Dessa forma, o protocolo inunda a rede com mensagens de interesse, exaurindo a capacidade do *buffer* rapidamente e levando ao comportamento mostrado.

As transmissões dos pacotes de interesse estão sujeitas aos critérios de cada protocolo, portanto, observa-se na Figura 4(b) que as curvas de envio são bem semelhantes as de criação de interesses. O objetivo desde gráfico é complementar a observação da Figura 4(c), de modo que se possa comparar a quantidade de requisições efetuadas e quantas delas foram atendidas.

A Figura 4(c) apresenta a quantidade de requisições originais, ou seja, feitas por nós que deram origem a pacotes de interesse, que foram satisfeitas. Uma vez que as curvas se apresentam de maneira muito semelhante as dos gráficos anteriormente tratados, complementamos a análise desta métrica com a Figura 4(d), que apresenta a relação

percentual entre interesses enviados e satisfeitos. O DIRESC apresenta um melhor desempenho em termos de satisfação em relação aos demais. Ainda que o PIFP se aproxime de nossa proposta a partir de certo ponto, ainda obtivemos um melhor resultado até 30000 segundos. Além disso, nosso esquema de priorização de envio garante que, no início dos experimentos, os interesses sejam enviados para nós com conteúdo na CS, uma vez que as informações de relacionamento social e de anúncios ainda estão sendo preparadas.

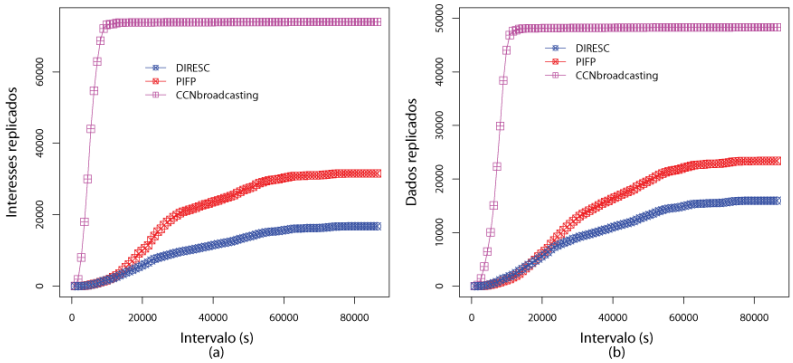


Figura 5. Carga total de pacotes de dados e interesses replicados na rede

As Figuras 5(a) e 5(b), por sua vez, apresentam o quanto de carga de mensagens é gerado na rede. Contabilizamos aqui a quantidade de réplicas de pacotes de dados e de interesse que foram feitas por retransmissões. Observa-se por meio deste gráfico que o DIRESC onera muito menos a rede em termos de sobrecarga em relação aos demais protocolos. Isto acontece devido ao esquema de priorização de envio, que, no caso dos interesses, antepõe nós que possuam o conteúdo ou qualquer informação sobre ele e, em último caso, privilegia nós "amigos", mesmo que isso também auxilie a recuperação de conteúdo e em reduzir o seu atraso; já no caso dos dados, a priorização privilegia qualquer nó interessado em obter o conteúdo, deixando para última instância nós que apenas possam fazer esta entrega. Assim, dispondo nós candidatos a serem retransmissores no último grau do critério de priorização, reduzimos o número total de réplicas na rede.

Nossa última métrica a ser analisada, o atraso de recuperação de conteúdo, compreende a média dos intervalos de tempo cumulativos obtidos desde o momento do envio de um pacote de interesse até o instante no qual o nó requisitante original recebe o pacote de dados que o satisfaça. Em vista das muitas replicações de mensagens e, portanto, do elevado preenchimento do *buffer*, os nós que operam por meio de CCN *broadcasting* estão mais propensos a recusar o recebimento de mensagens por falta de espaço neste recurso.

Devido a isto, podemos observar na Figura 6, um maior atraso (CCN *broadcasting*) até que os nós solicitantes possam receber os conteúdos requisitados, já que tanto interesses quanto dados podem ser largamente rejeitados no caminho entre origem e destino. Por sua vez, além do problema identificado de convergência de pacotes até o destino por meio de um único critério de envio, por considerar apenas o caminho reverso de envio dos pacotes de interesses para entregar os pacotes de dados, o PIFP ignora o fato de que contatos possam não se repetir, o que em DTN é muito provável, e, portanto, o conteúdo necessite ser entregue por outros caminhos. À vista disso, aumentam-se as chances de um nó ter de reexpresssar um determinado interesse, o que justifica o elevado número de réplicas de interesses da Figura 5(b). Assim, compreende-se que o PIFP apresenta um

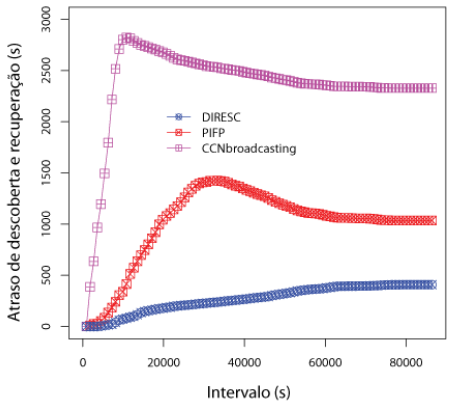


Figura 6. Atraso de recuperação de conteúdo

elevado atraso de recuperação de conteúdo por não propor um mecanismo de entrega diferenciado (ainda que o PIPF *Proative* se proponha a isto, mas caia no erro de inundação da rede pelo envio exacerbado de pacotes de dados). O DIRESC, por outro lado, por propor três possibilidades de envio de interesses para descoberta e três possibilidades de envio de dados para recuperação de conteúdo, e definir um esquema de ordenação por prioridades entre eles, aumenta significativamente as chances de um nó requisitante recuperar um dado. Neste contexto, não apenas os nós detentores de conteúdo ou de informação sobre ele, mas os nós de alto relacionamento social com os requisitantes são capazes de operar de modo socialmente cooperativo, em favor destes requisitantes tanto no processo de descoberta quanto na recuperação daquele conteúdo. Como observado na Figura 6, este esquema foi capaz de reduzir consideravelmente o tempo de atraso de recuperação de conteúdo em relação aos outros protocolos.

Para finalizar esta seção, a Tabela 2 sumariza de forma percentual os ganhos obtidos, e observados nos gráficos, do protocolo DIRESC primeiramente em relação à CCN *broadcasting* apresentando ganhos expressivos de no mínimo 45% e, em seguida, em comparação ao protocolo PIFP apresentando melhor desempenho, principalmente, nas métricas de carga e atraso.

Tabela 2. Ganhos percentuais observados

Protocolos	Taxa de satisfação(%)	Carga (nº dados)	Carga (nº interesses)	Atraso (s)
DIRESC x CCN *broadcasting*	45,3%	76,9%	85,29%	89,07%
DIRESC x PIFP	10,07%	30,43%	50,45%	73,33%

5. Conclusões

Neste artigo, propusemos o DIRESC, um protocolo para anúncio, descoberta e recuperação de dados em CCDTN baseado em cooperação social. Nossa proposta utiliza o cálculo de relacionamento social como fator principal para tomada de decisões. Ao apresentar esquemas diferenciados de envios de interesses e dados, nossa abordagem apresentou resultados significativamente melhores em comparação ao comportamento CCN padrão, isto é, em *broadcasting*, e ao protocolo PIFP, que é baseado em probabilidade de encontros entre nós e conteúdos.

Os resultados foram obtidos por meio de experimentos realizados no simulador ONE, no qual a proposta foi implementada, juntamente com a arquitetura CCN. As

medições apresentadas neste trabalho mostram que o DIRESC é capaz de acelerar o processo de obtenção de dados em redes CCDTN e alcançar uma elevada taxa de satisfação de requisições em comparação as outras propostas.

Como trabalho futuro, planejamos aprimorar o funcionamento do DIRESC para que realize gerenciamento de cache, isto é, seja capaz de lidar com políticas de substituição de dados na CS a fim de gerir e utilizar da melhor forma este recurso.

Referências

Amadeo, M., Campolo, C., Molinaro, A., and Ruggeri, G. (2014). Content-centric wireless networking: A survey. *Computer Networks*, 72:1–13.

Anastasiades, C., El Alami, W. E. M., and Braun, T. (2014). Agent-based content retrieval for opportunistic content-centric networks. In *WWIC*, pages 175–188.

Becker, V. D. et al. (2000). Epidemic routing for partially connected ad hoc networks. *Proceedings of Technique Report, Department of Computer Science, Duke University, Durham, UK*.

Conti, M., Boldrini, C., Kanhere, S. S., Mingozzi, E., Pagani, E., Ruiz, P. M., and Younis, M. (2015). From MANET to people-centric networking: Milestones and open research challenges. *Computer Communications*, 71:1–21.

Duarte, P., Macedo, J., Costa, A. D., Nicolau, M. J., and Santos, A. (2015). A Probabilistic Interest Forwarding Protocol for Named Data Delay Tolerant Networks. *7th AdhocNets 2015 San Remo, Italy, September 1-2, 2015 proceedings*, 155:94–107.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009). Networking named content. In *5th CoNEXT*, pages 1–12. ACM.

Junior, N. M., Campos, C. A. V., and Lucena, S. C. (2014). Uma proposta de solução para o problema de nós preferidos em protocolos oportunistas com base social. In *Anais do 32º SBRC*, pages 91–104.

Keränen, A., Ott, J., and Kärkkäinen, T. (2009). The ONE Simulator for DTN Protocol Evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA. ICST.

Kim, D. and Ko, Y.-B. (2015). A novel message broadcasting strategy for reliable content retrieval in multi-hop wireless content centric networks. *9th IMCOM '15*, pages 1–8.

Liu, X., Li, Z., Yang, P., and Dong, Y. (2016). Information-centric mobile ad hoc networks and content routing: A survey. *Ad Hoc Networks*, 0:1–14.

Lu, Y., Li, X., Yu, Y.-T., and Gerla, M. (2014). Information-centric delay-tolerant mobile ad-hoc networks. In *INFOCOM WKSHPS, IEEE Conference*, pages 428–433. IEEE.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Zhu, K., Li, W., Fu, X., and Zhang, L. (2015). Data routing strategies in opportunistic mobile social networks: Taxonomy and open challenges. *Computer Networks*, 93:183–198.

Explorando a afinidade de usuários para descarregamento de dados mais eficiente em redes celulares de pequeno porte

Adriana Viriato Ribeiro[1], **Leobino N. Sampaio**[1], **Artur Ziviani**[2]

[1]Programa de Pós-Graduação em Ciência da Computação (PGCOMP)
Instituto de Matemática – Universidade Federal da Bahia (UFBA)
Salvador – BA – Brasil

[2]Laboratório Nacional de Computação Científica (LNCC)
Petrópolis, RJ - Brasil

adrianavr@dcc.ufba.br, leobino@ufba.br, ziviani@lncc.br

Abstract. *Network-densification by Small Cell Networks has become the main alternative for mobile network operators when dealing with exponential traffic-growth. A Content-Centric Network can be an alternative to improve data offloading in this scenario. In this paper, we propose a user clustering strategy to take advantage of the similarity function that considers frequency and common-content requisitions to correlate users. Simulation results show that the proposed strategy can be adaptive to different scenarios and is able to increase hit ratio and minimize data offloading by 20% and 30%, respectively.*

Resumo. *A densificação de redes celulares através do uso de estações-base de pequeno porte é uma recente alternativa, adotada por operadoras de telefonia, para lidar com o crescente volume de tráfego de dados . No intuito de alcançar melhor taxa de descarregamento de dados, esse tipo de técnica tem sido associada às Redes Orientadas a Conteúdo. Neste artigo é proposta uma estratégia de agrupamento de usuários que explora a combinação da similaridade entre eles a partir dos conteúdos solicitados em comum e da frequência de solicitação a esses conteúdos. Os resultados de simulação demonstram a capacidade de adaptação da estratégia proposta a diferentes cenários e evidenciam melhoras de 20% em relação à taxa de acerto e de 30% ao descarregamento de dados.*

1. Introdução

A densificação de redes tem sido vislumbrada como uma das alternativas para as operadoras de telefonia móvel lidar com o crescente volume de tráfego de dados decorrente, principalmente, de serviços *streaming* [Aijaz et al. 2013, Bhushan et al. 2014]. Trata-se de uma maior oferta do serviço de telefonia a partir do aumento do número de antenas de transmissão para expandir a cobertura e prover descarregamento de dados (do inglês, *Data offloading*). Essas antenas geralmente são instaladas em pequenas estações-base (do inglês, *Small-Cell Base Station* – SCBS). As SCBS são constituídas por meio da implantação de equipamentos e tecnologias de transmissão de curto alcance, oferecendo, desta forma, redes celulares de pequeno porte (tradução adaptada do termo em inglês, *Small Cell Networks* – SCNs).[1]

[1]Na literatura, a expressão em inglês *Small cell* refere-se às chamadas "células pequenas" que englobam femto-células, pico-células e microcélulas. Essas redes aumentam a cobertura do serviço prestado pelas operadoras e são utilizadas pelos usuários na redução do congestionamento das macro-células.

As SCNs oferecem serviços complementares aos disponibilizados pelas macro estações-base (do inglês, *Macro Base Station* – MBS) com o propósito de aumentar a cobertura do serviço e atender às demandas de tráfego locais, melhorando os tempos de resposta da rede. Tais objetivos tem sido potencializados através das Redes Orientadas à Conteúdo (ROC) [Jacobson et al. 2009]. Propostas descritas na literatura sugerem o uso de ROCs em cenários de SCN para manter os conteúdos mais populares na SCBS e, consequentemente, mais próximos dos consumidores finais [ElBamby et al. 2014, Pantisano et al. 2015, Chen and Kountouris 2015, Hajri and Assaad 2016]. Dado que as políticas clássicas de gerenciamento de *cache*, tais como *Least Frequently Used* (LFU) e *Least Recently Used* (LRU), não levam em consideração as preferências individuais dos usuários, trabalhos mais recentes têm sugerido o uso de técnicas cientes de contexto [Chang et al. 2016] e de agrupamento com o objetivo de reunir usuários com base em características similares e associá-los às SCBS mais apropriadas [Ioannou and Weber 2016]. Dessa forma, espera-se que, através de agrupamentos, os *caches* consigam alcançar melhores taxas de acerto e descarregar a MBS [ElBamby et al. 2014, Pantisano et al. 2015].

Apesar do agrupamento apresentar ganhos em relação ao descarregamento de dados, a maioria dos trabalhos relatados na literatura [ElBamby et al. 2014, Pantisano et al. 2015] que recorrem a esta estratégia leva em consideração apenas a frequência dos conteúdos na definição dos grupos que serão associados às SCBS. O problema, contudo, é que a frequência dos conteúdos não relata de forma precisa o nível de afinidade entre os usuários, visto que, dois consumidores podem ter interesse apenas por um único conteúdo em comum e, ainda assim, serem considerados similares em decorrência da frequência de solicitações e da função de similaridade utilizada.

Este trabalho, portanto, apresenta uma estratégia que explora a formação de grupos de usuários similares em cenários de uso de *caches* em SCN. A técnica de agrupamento proposta se baseia nos interesses dos consumidores de conteúdos e visa obter melhores taxas de descarregamento de dados na MBS através do aumento da taxa de acerto nas SCBS. A estratégia proposta busca identificar grupos que retratem melhor o nível de afinidade entre os usuários considerando, não somente a frequência de solicitações, mas também a quantidade de conteúdos solicitados em comum. Os resultados obtidos demonstraram que quando a similaridade leva em consideração tanto a quantidade de conteúdos em comum quanto a frequência, as taxas de acerto e descarregamento são superiores a quando apenas um fator é utilizado. Além disso, a função de similaridade proposta pode ser adaptada a diferentes cenários, uma vez que a influência da frequência e do conteúdo pode ser ajustada a partir da atribuição de um peso às diferentes partes da equação.

O restante deste artigo está organizado da seguinte forma. A Seção 2 discute brevemente as principais propostas que fazem uso de técnicas de agrupamento em cenários de SCN. A Seção 3 apresenta a estratégia adotada. A Seção 4 descreve o ambiente de avaliação adotado e a Seção 5 analisa os principais resultados experimentais obtidos. Por fim, a Seção 6 conclui o artigo e discute possíveis trabalhos futuros.

2. Gerenciamento de *Cache* em Redes Orientadas à Conteúdo

As Redes Orientadas à Conteúdo [Jacobson et al. 2009] tem como característica a desvinculação entre um conteúdo e seu localizador, com o objetivo de focar em *qual* conteúdo o usuário quer, em vez de *onde* o conteúdo está localizado. Essa abordagem

apresenta novas estruturas que dão suporte ao tráfego de conteúdo nomeado: a *Pending Interest Table* (PIT) armazena as requisições que ainda não foram atendidas, a *Forwarding Information Base* (FIB) mantém as informações de encaminhamento e a *Content Store* é uma estrutura de armazenamento similar a um *cache*. A utilização do *cache* nos próprios equipamentos de rede é um dos diferenciais dessa abordagem e essa característica tem sido utilizada em diferentes contextos.

Existem diversas políticas de gerenciamento de *cache* [Ioannou and Weber 2016] com diversas adaptações para se adequar aos diferentes cenários , no entanto, essas estratégias geralmente não levam em consideração as preferências individuais dos usuários. Desta forma, começaram a ser criadas estratégias que fizessem agrupamentos entre usuários com características similares para que os *caches* conseguissem alcançar melhores taxas de acerto e descarregamento.

Em [ElBamby et al. 2014] é descrita uma solução que faz uma associação entre usuários e *caches* de SCBS. A solução visa melhorar o descarregamento de dados na MBS e diminuir o tempo de resposta ao usuário. A associação entre usuário e SCBS é feita de acordo com a similaridade entre eles, que é calculada utilizando a lei do cosseno em relação à frequência dos conteúdos requisitados. Após a associação, os *caches* das SCBS são atualizados de acordo com um método de aprendizado por regressão também proposto no artigo. A estratégia conseguiu minimizar o atraso em 42% e 27% e obteve um ganho de 280% e 90% quando comparado com o agrupamento feito de forma aleatória e com esquemas sem agrupamentos.

[Pantisano et al. 2015] propõe um algoritmo de associação entre usuários e *caches* com a intenção de fazer uma melhor alocação da largura de banda, aumentar QoS do usuário e reduzir retransmissões. O cálculo de similaridade se baseia em técnicas de Filtro Colaborativo e é realizado levando em consideração a frequêncida da média de requisições recebidas por cada SCBS, da média de requisições para cada conteúdo e da média de requisições de cada usuário para todos os conteúdos. A intenção é explorar as relações entre as requisições de conteúdo e os usuários, levando em consideração a probabilidade de um conteúdo ser requisitado por outro usuário. Os resultados demonstraram ganhos em relação à alocação de largura de banda e taxa de acerto com melhoras de até 25%.

3. Estratégia de Agrupamento por Interesses Similares

A solução desenvolvida visa aumentar a taxa de acerto nas SCBS para diminuir a sobrecarga na MBS. Para tanto, a solução conta com o uso de agrupamentos baseados na similaridade de usuários em relação aos seus interesses. Nesta seção serão descritos a estrutura da rede e os tipos de pacotes trafegados, a função de similaridade, o algoritmo de agrupamento e as formas de comunicação existentes.

3.1. Estrutura da Rede e Tipos de Pacotes

O cenário é composto por três entidades distintas:

- a MBS é a estação principal, capaz de prover todos os conteúdos que estão disponíveis na rede;
- a SCBS tem capacidade de armazenamento limitada e é utilizada para armazenar réplicas de conteúdo mais próximas dos usuários;

- os dispositivos móveis (DM) são os clientes/assinantes de conteúdo que fazem requisições através do envio de pacotes de interesse.

A comunicação entre os equipamentos é feita através do envio de pacotes. os pacotes de interesse tem a finalidade de requisitar conteúdos nomeados, os pacotes de dados também são nomeados e comportam o conteúdo propriamente dito, e os pacotes de controle são pacotes de dados cujo conteúdo é utilizado na manutenção da rede.

Cada entidade é responsável por manipular esses pacotes de acordo com a sua funcionalidade. Os dispositivos móveis são responsáveis pelas requisições de dados através do envio de pacotes de interesse. Cada dispositivo contém uma tabela de requisições que é composta pelo nome do conteúdo e a frequência em que foi solicitado. Desta forma, toda vez que um pacote de interesse é enviado, a tabela de requisições é atualizada. Essa tabela é enviada à MBS através de um pacote de controle e é utilizada para realização do cálculo de similaridade e agrupamento entre os dispositivos.

As principais funções da MBS são as de processar pacotes de interesse e controle. Ao receber um pacote de interesse, a MBS faz uma busca no *cache* para encontrar o pacote de dados referente àquele interesse e enviá-lo ao nó solicitante. A MBS é capaz de prover todos os conteúdos disponíveis na rede. Além disso, é responsável por realizar o cálculo de similaridade e a formação dos agrupamentos, conforme a Seção 3.2.

A SCBS pode receber e solicitar pacotes de interesse e dados. Ao receber um pacote de interesse, verifica no *cache* se existe um pacote de dados relacionado àquele conteúdo, se existir, encaminha o pacote de dados para o nó solicitante. Caso contrário, o nome do conteúdo e o nó solicitante são adicionados à PIT. Em seguida, a SCBS solicita o conteúdo à MBS e, ao recebê-lo, adiciona no seu *cache* e envia para o nó requisitante. Vale ressaltar que a ROC faz agregação de requisições, portanto, se mais de um nó solicitar o mesmo conteúdo, apenas uma entrada na PIT é adicionada com o nome do conteúdo e a lista de endereços dos nós solicitantes. Desta forma, o pacote de interesse é enviado uma única vez através da rede e o pacote de dados é replicado para todos os nós assinantes.

De acordo com as diferentes funcionalidades de cada entidade, é possível observar que o envio de pacotes de dados pode ocorrer de três maneiras distintas, como pode ser observado na Figura 1:

- a MBS envia o conteúdo diretamente para o DM;
- a SCBS tem o conteúdo armazenado em *cache* e envia para o DM;
- a SCBS não tem o conteúdo armazenado e a MBS envia o conteúdo para ela.

Cada uma dessas situações ocorre de acordo com a configuração da rede em determinado instante. Antes da formação dos grupos, todos os pacotes de dados são enviados pela MBS. Após a formação dos agrupamentos, os usuários são servidos pelas SCBS. A MBS só enviará conteúdos para a SCBS quando a SCBS não tiver o conteúdo solicitado armazenado em *cache*. Em uma ROC, um pacote de dados só é enviado em resposta a um pacote de interesse, logo, toda vez que a SCBS necessitar de um conteúdo que não existe no seu *cache*, deverá enviar uma requisição através do envio de um pacote de interesse.

Como foi citado anteriormente, além dos pacotes de interesse e de dados, existem os pacotes de controle. Os pacotes de controle são utilizados para realizar a formação dos agrupamentos. Os primeiros pacotes de controle enviados contém as tabelas de

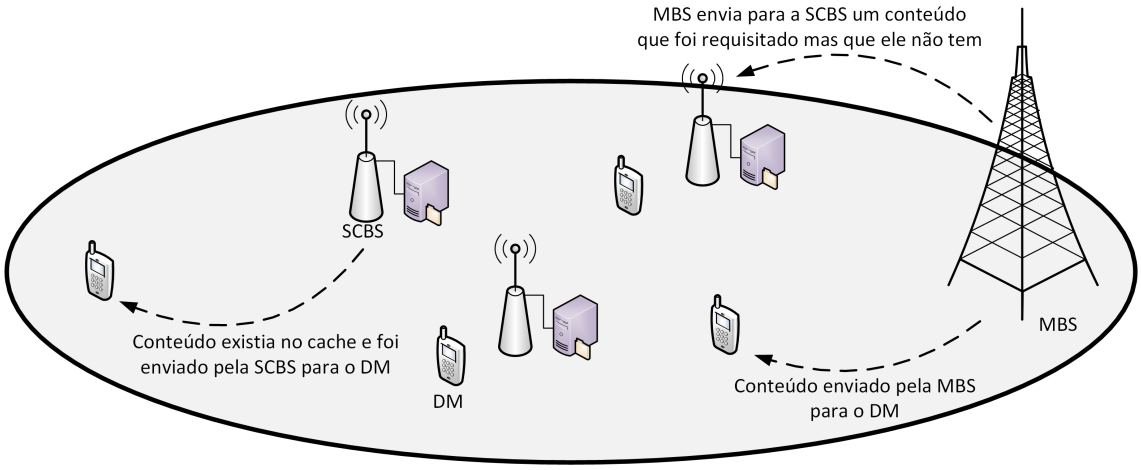


Figura 1. Estrutura da SCN utilizada.

requisições dos usuários e são utilizados como insumo para o cálculo de similaridade e formação dos agrupamentos.

3.2. Função de Similaridade

A função de similaridade tem o objetivo de quantificar a semelhança entre diferentes usuários de acordo com seus interesses.

A similaridade entre os usuários é medida através da Equação 1. A primeira parte representa o Índice de Jaccard [Jaccard 1912], comumente utilizado para medir a similaridade através das características semelhantes e distintas entre dois indivíduos. Ela foi utilizada para medir a similaridade de acordo com os conteúdos em comum que dois nós (i, j) tem interesse. Já a segunda parte é calculada utilizando a lei de similaridade do cosseno em relação à frequência de requisições de conteúdos em comum.

Desta forma, C_i e C_j representam os conjuntos de conteúdos que foram solicitados por i e por j, respectivamente, e F_i e F_j representam a frequência, em i e em j, em que os conteúdos comuns foram requisitados. O β define um peso entre as partes da equação e permite que seja avaliado o impacto delas no cálculo de similaridade.

$$S_{i,j} = \beta\left(\frac{|C_i \cap C_j|}{|C_i \cup C_j|}\right) + (1-\beta)\left(\frac{F_i \cdot F_j}{\|F_i\| \left\|F_j\right\|}\right) \quad (1)$$

Vale ressaltar, que não estamos avaliando a similaridade entre usuário e *cache*, nem entre dois conteúdos, mas a similaridade entre usuários de acordo com seus interesses por conteúdo. Além disso, neste trabalho, o cenário "com similaridade" representa o uso da função de similaridade na formação dos agrupamentos. Enquanto no cenário "sem similaridade" os membros são adicionados de acordo com sua identificação. Por exemplo, os nós de 1 a 10 formam o primeiro agrupamento, os nós de 10 a 20 formam o segundo, e assim por diante. Isso não quer dizer que não exista similaridade entre os usuários, mas que a similaridade não está sendo levada em consideração na formação dos grupos.

3.3. Algoritmo de Agrupamento

O algoritmo de agrupamento (Algoritmo 1) é executado na MBS e tem a função de estabelecer as associações entre nós que tenham interesses similares. Esse trabalho avalia o efeito de apenas uma execução do algoritmo.

Os agrupamentos são determinados utilizando as tabelas de requisições dos nós como insumo. A partir de um determinado instante de tempo t, cada usuário envia uma única vez um pacote de controle contendo a sua tabela de requisições. Cada tabela de requisição recebida pela MBS é adicionada a uma matriz NxC, sendo N a quantidade de nós da rede e C a quantidade de conteúdos disponíveis. Ao receber a tabela de requisição, a MBS verifica o ID do nó que enviou a requisição e adiciona na linha da matriz referente àquele ID. Depois de receber as requisições de todos os nós da rede, é inicializado o cálculo da similaridade. Para saber quando todas as requisições foram recebidas, foi utilizado um contador inicializado em zero e incrementado até a quantidade de nós.

Algoritmo 1: Algoritmo de Agrupamento

```
   Entrada: Matriz NxC
1  início
2      para cada i ∈ N faça
3          para cada j ∈ N faça
4              para cada w ∈ C faça
5                  Calcula os valores de C_i ∩ C_j , C_i ∪ C_j , F_i · F_j e ||F_i|| ||F_j||;
6              fim
7              Calcula a similaridade entre i e j;
8              Constrói uma matriz NxN com a similaridade dos nós;
9          fim
10     fim
11     para cada z ∈ s faça
12         Aplica o método de ordenação por seleção em cada linha da matriz;
13         Calcula a média aritmética entre os k vizinhos mais similares de cada
             nó;
14         Identifica o nó com maior média de similaridade;
15         Forma um agrupamento com o nó com maior média e seus k vizinhos
             mais similares;
16         Envia mensagem de controle para os nós do agrupamento formado
             identificando o servidor de distribuição que eles devem se associar;
17         Zerar os valores de similaridade dos nós que já foram agrupados;
18     fim
19 fim
   Saída: s agrupamentos, sendo s a quantidade de SCBS
```

O cálculo da similaridade é realizado par a par entre cada nó da rede, gerando uma matriz NxN. O índice de cada linha da matriz corresponde ao ID de um dos nós que compõe a rede e a linha representa a similaridade entre um nó e os demais. A linha 1, por exemplo, equivale à similaridade entre o nó 1 e todos os outros nós da rede $\{1, 2, 3, ..., N\}$. Além disso, a diagonal principal dessa matriz é uma diagonal nula, pois representa a similaridade entre um nó e ele mesmo.

Após a formação da matriz contendo a similaridade entre todos os nós da rede, é feita uma ordenação decrescente entre os elementos de cada linha da matriz, para que os nós sejam ordenados de acordo com a similaridade. Cada elemento da matriz é uma *struct* formada pelo ID do nó e pela similaridade, isso garante que, mesmo após a ordenação, sejam mantidos os IDs dos nós que mudaram de posição.

Com a matriz de similaridade ordenada, o próximo passo é calcular a média de similaridade entre um nó e os seus k vizinhos mais próximos (do inglês, *k Nearest Neighbor*), obtendo assim a média de similaridade de cada nó da rede. Em seguida, o nó com maior média é identificado e inicia-se a formação do agrupamento.

O agrupamento é formado pelo nó que tem a maior média e os seus k primeiros vizinhos, que representam os vizinhos com maior similaridade. O valor de k é calculado por $(N/S)-1$. Sendo que N é a quantidade de nós, S é a quantidade de SCBS e a subtração por 1 é justificada pelo fato de que o primeiro nó do agrupamento já foi determinado (nó com maior média de similaridade). Esse cálculo garante que os nós sejam distribuídos de maneira uniforme entre as SCBS. A distribuição uniforme foi feita com o intuito de equilibrar a quantidade de usuários por SCBS, para que a quantidade de usuários por antena não interferisse nos resultados do efeito da equação de similaridade. Por exemplo, supõe-se que uma antena com dois usuários conectados teria uma taxa de acerto maior do que uma antena com 20 nós conectados, ainda que fossem bastante similares.

Ao identificar os membros do agrupamento, a MBS envia um pacote de controle para cada usuário identificando a SCBS que ele deve se associar. A partir desse momento, os DM fazem requisições apenas para as SCBS, deixando de requisitar à MBS.

No Algoritmo 1 a complexidade do cálculo de similaridade é de $O(CN^2)$ enquanto a realização do agrupamento tem complexidade $O(N^3)$. Sendo C a quantidade de conteúdos e N a quantidade de usuários.

3.4. Formas de Comunicação

Após a formação dos agrupamentos, existem duas possibilidades de comunicação. Partindo do pressuposto de que um DM envia um conteúdo nomeado para a SCBS1 e o único nó da rede que possui o pacote de dados associado a esse conteúdo é a MBS, é possível que a comunicação ocorra de duas maneiras:

- Comunicação Direta, conforme Figura 2: ao receber um pacote de interesse de um dispositivo móvel (1), SCBS1 verifica a sua *cache* (2) e percebe que não existe um conteúdo relacionado ao interesse. Desta forma, é adicionada uma entrada na PIT (3) e o pacote de interesse é encaminhado diretamente a MBS (4). A MBS acessa o *cache* para obter o conteúdo solicitado (5,6) e envia o pacote de dados para SCBS1 (7), SCBS1 armazena o pacote de dados no seu *cache* (8), apaga a entrada da PIT (9) e encaminha o pacote de dados para o dispositivo móvel solicitante (10);
- Comunicação Indireta, conforme Figura 3: o DM envia um pacote de interesse para SCBS1 (1), como SCBS1 não tem o conteúdo referente ao interesse, ela encaminha o interesse à SCBS2 (2) e a SCBS2 realiza o mesmo processo de verificação e encaminhamento (3). Ao chegar na SCBS3, o pacote de interesse já percorreu todos os nós da rede daquele nível, portanto, a única entidade que pode resolver

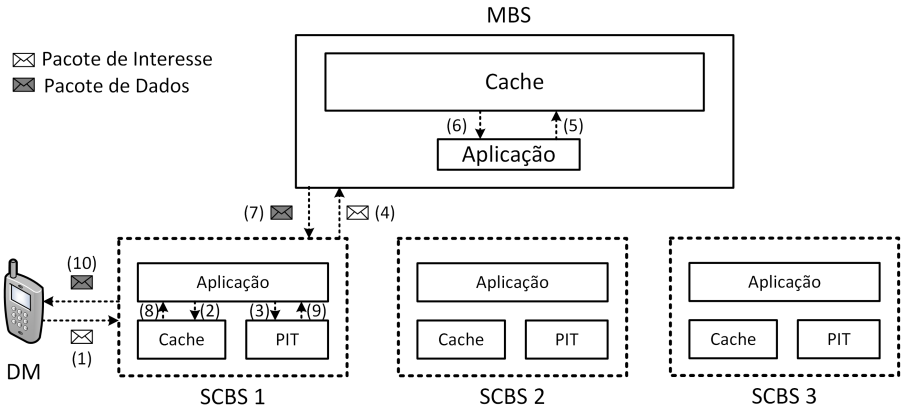


Figura 2. Comunicação Indireta.

o interesse é a MBS. Logo, o servidor SCBS3 envia o pacote de interesse para a MBS (4) e a MBS envia o pacote de dados diretamente para SCBS1 (5). Isso é possível porque mesmo o pacote sendo encaminhado entre as SCBS, o endereço da primeira SCBS solicitante é mantido nas mensagens. Por fim, a SCBS1 encaminha o pacote de dados para o DM (6).

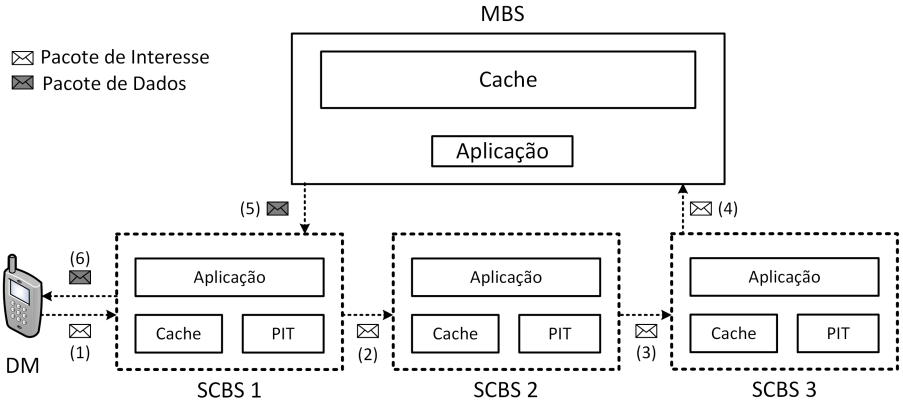


Figura 3. Comunicação Direta.

4. Experimentos

Para realizar os experimentos, foi necessário determinar cenários de avaliação, métricas, fatores e parâmetros. Todos esses aspectos serão abordados nessa seção.

4.1. Cenários, Métricas, Fatores e Parâmetros

Os fatores utilizados podem ser vistos na primeira coluna da Tabela 1. A similaridade representa a utilização da equação de similaridade e o β representa o peso usado na equação. A comunicação retrata a direção no encaminhamento dos pacotes, o tamanho do *cache* indica espaço disponível para armazenamento, e o α representa o parâmetro da Distribuição de *Zipf*, que é responsável por definir a popularidade dos conteúdos.

De acordo com esses fatores foram definidos três cenários de avaliação: o Grupo 1 compara o impacto do parâmetro β e do Tamanho da *Cache* em relação à taxa de acerto;

Tabela 1. Fatores e Cenários de Avaliação.

Fatores	**Grupo 1**	**Grupo 2**	**Grupo 3**
Similaridade	Sim - Não	Sim - Não	**Sim - Não**
Comunicação	Direta	Direta	**Indireta - Direta**
Tamanho do *Cache*	**10 - 20 - 30**	20	20
β (Funç. Similaridade)	**0 - 0.5 - 1**	**0 - 0.5 - 1**	0.5
α(Distr. Zipf)	1	**0.6 - 0.8 - 1**	1

o Grupo 2 avalia o impacto do parâmetro Zipf e do parâmetro β na taxa de acerto; e, por fim, o Grupo 3 analisa a utilização da similaridade e da hierarquia em relação à taxa de acerto e ao descarregamento de dados.

Os experimentos foram avaliados de acordo com duas métricas: taxa de acerto na SCBS e descarregamento de dados na MBS. O descarregamento de dados visa medir a quantidade de requisições que deixaram de ser feitas na MBS e pode ser representado pela Equação 2, na qual *rMax* representa os interesses que foram requisitados quando todas as requisições eram feitas diretamente na MBS e *rMin* são as requisições recebidas após a formação dos agrupamentos.

$$off = 100 \times \frac{rMax - rMin}{rMax} \tag{2}$$

Já a taxa de acerto mede a quantidade de vezes que uma SCBS recebeu um pacote de interesse e tinha o pacote de dados no *cache*. Essa taxa é medida de acordo com a Equação 3, na qual *match* representa os interesses que foram resolvidos ao chegar na SCBS e *iRec* o total de todos os interesses que a SCBS recebeu. A taxa de acerto total é calculada através da média aritmética da taxa de acerto das três SCBS.

$$hr = 100 \times \frac{match}{iRec} \tag{3}$$

4.2. Ambiente de Simulação

A solução proposta foi implementada no simulador OMNeT++ utilizando o *framework* INET, conforme parâmetros da Tabela 2.

A área simuladada é de 600x800m onde estão distribuídos 30 dispositivos móveis que se movimentam de acordo com o modelo de mobilidade *Mass Mobility* e requisitam conteúdo seguindo o modelo de Poisson. O meio físico sem fio tem propagação do tipo *Constant Speed Propagation*, frequência de 2.4GHz, perda do tipo *Free Space Path Loss* e ruído de -110dBm, todos esses parâmetros já vieram especificados no INET. Foi criada uma MBS que alcança toda área de simulação e foram criadas três SCBS cujo alcance é de 300m. Na simulação, cada SCBS é associada a 10 dispositivos móveis, de forma imperativa, de acordo com a estratégia utilizada. Existem 100 conteúdos disponíveis na rede e, para determinar perfis diferentes para os usuários, foi selecionado um conjunto de conteúdo para cada dispositivo. Para cada subconjunto de conteúdo é aplicado o modelo de popularidade de *Zipf*. O tamanho do *cache* da SCBS varia entre 10% e 30% da

Tabela 2. Parâmetros da Simulação.

Parâmetro	Valor
Quantidade de dispositivos móveis na rede	30
Quantidade de MBS	1
Quantidade de SCBS	3
Quantidade de conteúdos disponíveis	100
Política de descarte de *cache*	LFU
Modelo de Mobilidade	*MassMobility*
Tipo de Propagação	*ConstantSpeedPropagation*
Frequência	2.4GHz
Perda	*FreeSpacePathLoss*
Ruído	-110dBm

quantidade de conteúdos e a estratégia de descarte utilizada é a LFU. O parâmetro β, que determina o peso na equação de similaridade, é variado entre 0 e 1 e são comparados cenários com comunicação direta e indireta. Cada experimento foi executado 10 vezes com um Intervalo de Confiança de 95%.

5. Análise dos Resultados

Nesta seção são apresentados os resultados e as análises dos diferentes cenários em relação à taxa de acerto e de descarregamento de dados de acordo com as variações de fatores apresentadas na Tabela 1.

5.1. Grupo 1: Relação entre Tamanho do *Cache*, β e Taxa de Acerto

A Figura 4 expõe o comportamento da taxa de acerto em relação ao uso da função de similaridade e da variação do tamanho do *cache*. Em relação ao uso da equação de similaridade, para todos os tamanhos de *cache* (10, 20, 30), é possível observar que quando apenas a quantidade de conteúdos em comum é levada em consideração (β = 1), a taxa de acerto é superior a quando apenas a frequência é utilizada (β = 0). Isso pode ser justificado devido ao fato de que ao contabilizar apenas a frequência, é possível que dois usuários sejam considerados bastante similares mesmo tendo um único conteúdo em comum. Por exemplo, quando dois nós tem um conteúdo em comum, ainda que requisitado uma única vez, de acordo com a função da similaridade do cosseno, esses dois nós terão similaridade igual a 1. Dessa forma, como o Índice de Jaccard leva em consideração a relação entre os conteúdos requisitados em comum e todos os conteúdos requisitados pelos dois nós, os agrupamentos baseados nessa relação de similaridade conseguem alcançar resultados superiores. Apesar disso, os melhores resultados foram obtidos quando ambos (β = 0.5) foram utilizados, evidenciando a importância de uma equação de similaridade que considere esses dois fatores.

Vale ressaltar que ao levar em consideração apenas a frequência de conteúdos (β = 0) calculada pela similaridade do cosseno, não estamos replicando o cenário descrito em [ElBamby et al. 2014], visto que o autor calcula similaridade usuário-*cache* e neste trabalho é calculada a similaridade usuário-usuário.

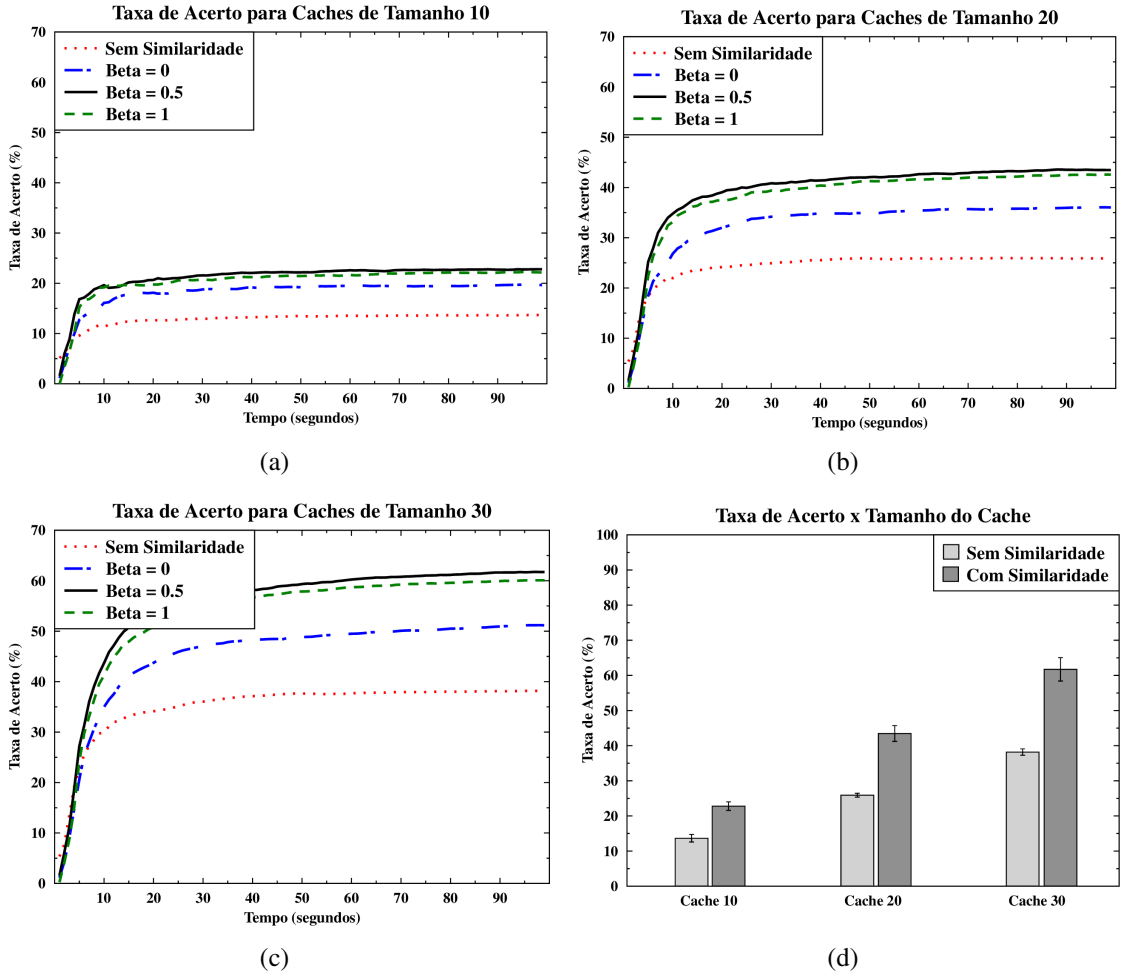


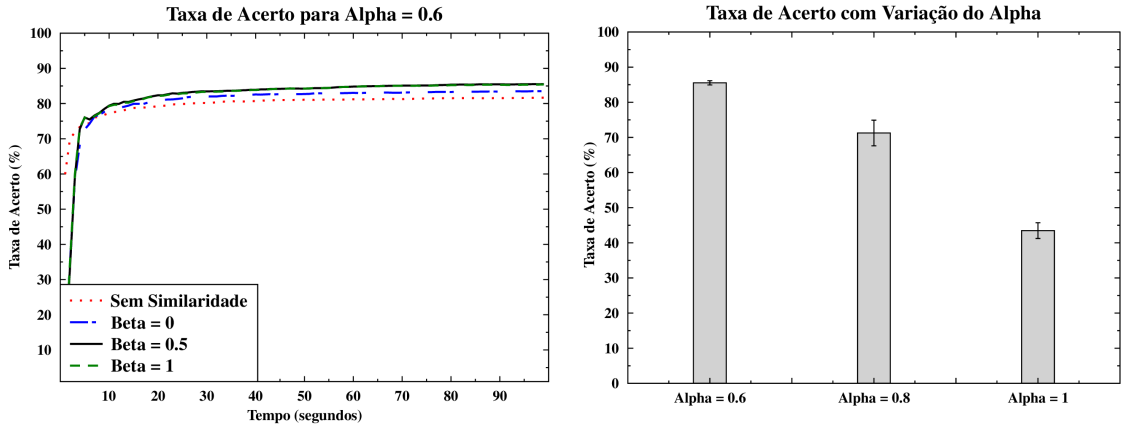
Figura 4. Relação entre o Tamanho do *Cache* e a Taxa de Acerto.

A distinção entre as taxas de acerto dos gráficos das Figuras 4(a) 4(b) 4(c) se dá devido ao aumento do tamanho do *cache*. No entanto, ao relacionar os tamanhos do *cache* 10 e 20, conforme a Figura 4(d), observa-se que a taxa de acerto obtida sem similaridade com um *cache* = 20 alcança valores próximos ao melhor valor obtido com uso de similaridade em um *cache* = 10. Ao se comparar os *caches* de tamanho 20 e 30, percebe-se que com o uso da similaridade em um *cache* = 20 é possível obter valores superiores ao cenário sem similaridade em um *cache* = 30. Isso evidencia que a utilização de agrupamentos levando em consideração a similaridade dos usuários torna, de fato, o gerenciamento de *cache* mais eficiente e possibilita economia de recursos, visto que os mesmos valores de taxa de acerto podem ser obtidos com *caches* de tamanho inferior.

5.2. Grupo 2: Relação entre o parâmetro α da Lei de Zipf e a Taxa de Acerto

A Figura 5(a) evidencia que mesmo com diferentes valores para o parâmetro Zipf, o comportamento da função de similaridade permanece o mesmo: os resultados com valor de $\beta = 0.5$ ou $\beta = 1$ são superiores a quando considera-se apenas a frequência. No entanto, é possível observar na Figura 5(b) que, apesar do comportamento da função de similaridade permanecer o mesmo, os valores de taxa de acerto alcançados são distintos. Quanto menor o valor do parâmetro Zipf, maior a taxa de acerto obtida. Isso acontece porque

quando o parâmetro Zipf é menor, há uma maior quantidade de conteúdos populares que, consequentemente, são requisitados mais vezes. Como a política de descarte utilizada é a LFU, os conteúdos mais populares ficam armazenados por mais tempo, satisfazendo as requisições e culminando no aumento da taxa de acerto.

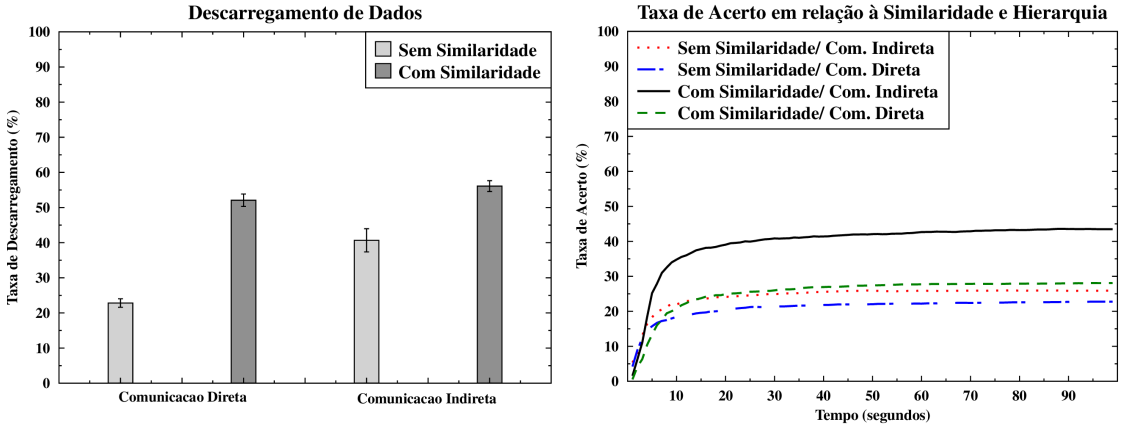


(a) (b)

Figura 5. Efeito do parâmetro Zipf na Taxa de Acerto.

5.3. Grupo 3: Relação entre Similaridade e Comunicação na Taxa de Acerto e no Descarregamento de Dados

Na Figura 6(a) é possível observar o comportamento entre similaridade e comunicação em relação à taxa de descarregamento. A taxa de descarregamento de aproximadamente 22% obtida no cenário sem similaridade e com comunicação direta, decorre da adição das três SCBS. O cenário com comunicação indireta e com similaridade consegue obter resultados superiores ao cenário com comunicação direta e sem similaridade, evidenciando que a utilização de agrupamentos influencia mais no descarregamento do que a forma de comunicação. Os melhores resultados foram obtidos com comunicação indireta e similaridade.



(a) (b)

Figura 6. Influência da Similaridade e da Hierarquia na Taxa de Acerto e de Descarregamento de Dados.

Em relação a avaliação da taxa de acerto nos cenários com e sem similaridade e comunicação direta e indireta, é possível observar, de acordo com a Figura 6(b), que os cenários de comunicação direta obtém taxa de acerto superior aos cenários de comunicação indireta. Isso acontece porque quando a comunicação é indireta, as SBCS recebem pacotes de interesse que não pertencem aos usuários do grupo que está associado a elas, portanto, a probabilidade delas terem o conteúdo armazenado em *cache* é menor. Apesar disso, o grupo de usuários associado a cada SCBS não sofre nenhum impacto decorrente dessa diminuição da taxa de acerto, visto que não há substituição em *cache* dos conteúdos que foram solicitados por outra SCBS. No entanto, há um aumento no tempo de resposta ao usuário decorrente da comunicação indireta, visto que pode aumentar a quantidade de saltos para alcançar o conteúdo.

Na Figura 7 foi feito um experimento com apenas 30 conteúdos, no qual todos os usuários requisitam os conteúdos disponíveis com diferentes frequências, logo, apenas a parte da equação que representa a similaridade do cosseno irá refletir a similaridade dos usuários. Assim, é possível observar que, contabilizando apenas os conteúdos em comum ($\beta = 1$), a taxa de acerto é inferior a quando a frequência é considerada ($\beta = 0$). Isso acontece porque como todos os usuários requisitam todos os conteúdos, o Índice de Jaccard não consegue detectar a similaridade entre os usuários, pois os conjuntos de conteúdo serão sempre iguais. Desta forma, ao utilizar $\beta = 1$, os resultados são semelhantes a um cenário sem similaridade.

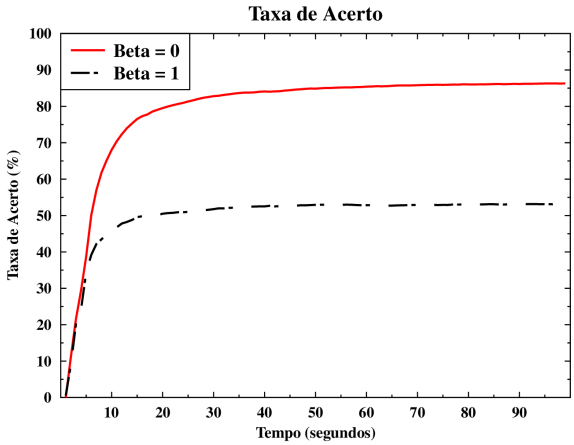


Figura 7. Representação da adaptatividade da função de similaridade.

Também já foi exposto que quando dois nós tem um único conteúdo em comum solicitado uma única vez, de acordo com a similaridade do cosseno, esses nós são caracterizados com grau de similaridade máximo. Dessa forma, fica evidente a importância da utilização de ambas as partes da função de similaridade (Equação 1) durante a formação dos agrupamentos, tornando o cálculo de similaridade adaptativo às características do ambiente e capaz de superar lacunas de cada indicador de similaridade.

6. Conclusão e Trabalhos Futuros

Os resultados obtidos demonstraram que, ao utilizar a função de similaridade proposta, foi possível obter valores de taxa de acerto superiores ao cenário aleatório mesmo com tamanho de *cache* inferior, evidenciando a economia de recursos e aumento da taxa de acerto.

Ao utilizar a comunicação indireta há um aumento no tempo de resposta ao usuário e degradação da taxa de acerto nas SCBS. Apesar disso, os melhores resultados para descarregamento de dados na MBS envolvem a comunicação indireta e o uso de similaridade.

Como trabalhos futuros, pretende-se aumentar a quantidade de conteúdos e de nós, também levando em consideração os nós que não se conectam a todas as SCBS. Além disso, será avaliada a entrada e saída de nós no ambiente para verificar a necessidade de recalcular a similaridade entre os usuários e a distribuição dos nós nas SCBS não será realizada uniformemente.

Agradecimentos

Os autores agradecem o apoio da CAPES e do CNPq.

Referências

Aijaz, A., Aghvami, H., and Amani, M. (2013). A Survey on Mobile Data Offloading: Technical and Business Perspectives. *IEEE Wireless Communications*, 20(2):104–112.

Bhushan, N., Li, J., Malladi, D., Gilmore, R., Brenner, D., Damnjanovic, A., Sukhavasi, R. T., Patel, C., and Geirhofer, S. (2014). Network Densification: The Dominant Theme for Wireless Evolution into 5G. *IEEE Communications Magazine*, 52(2):82–89.

Chang, Z., Gu, Y., Han, Z., Chen, X., and Ristaniemi, T. (2016). Context-Aware Data Caching for 5G Heterogeneous Small Cells Networks. In *Communications (ICC), 2016 IEEE International Conference on*, pages 1–6. IEEE.

Chen, Z. and Kountouris, M. (2015). Cache-enabled Small Cell Networks with Local User Interest Correlation. In *IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 680–684.

ElBamby, M. S., Bennis, M., Saad, W., and Latva-Aho, M. (2014). Content-aware User Clustering and Caching in Wireless Small Cell Networks. In *11th International Symposium on Wireless Communications Systems (ISWCS)*, pages 945–949.

Hajri, S. E. and Assaad, M. (2016). Caching Improvement Using Adaptive User Clustering. *arXiv preprint arXiv:1605.09602*.

Ioannou, A. and Weber, S. (2016). A Survey of Caching Policies and Forwarding Mechanisms in Information-Centric Networking. *IEEE Communications Surveys & Tutorials*, pages 1553–877.

Jaccard, P. (1912). The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009). Networking Named Content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, pages 1–12.

Pantisano, F., Bennis, M., Saad, W., and Debbah, M. (2015). Match to Cache: Joint User Association and Backhaul Allocation in Cache-aware Small Cell Networks. In *IEEE International Conference on Communications (ICC)*, pages 3082–3087.

Um Modelo de Rede Centrada na Informação Resiliente a Ataques de Negação de Serviços por Inundação de Interesses

Nilton Flávio S. Seixas[1], **Adriana Viriato Ribeiro**[1], **Leobino N. Sampaio**[1]

[1]Programa de Pós-graduação em Ciência da Computação (PGCOMP)
Instituto de Matemática – Universidade Federal da Bahia (UFBA)
Salvador – BA – Brasil

nfsseixas@dcc.ufba.br, adrianavr@dcc.ufba.br, leobino@ufba.br

Abstract. *In denial of services attacks by IFA (Interest Flooding Attack) the routers of caching network become unable to forward or receive legitimate packets, due to an exhaustion of memory reserved to PIT, occupied by an excessive number of false interests. In this work, it is proposed a model of Information Centric Network that sugests a cooperation between providers of content and routers to provide resilience to the practice of IFA. The cooperation facilitates the discrimination of illegitimate packets, making the model reaching more efficiency in the processes of detection and mitigation. Obtained results throught the simulation of caching network on the backbone of Internet2 demonstrated the effectiveness of proposed model.*

Resumo. *Em ataques de negação de serviço do tipo IFA (Interest Flooding Attack), os roteadores de uma rede de cache ficam inaptos para encaminhar ou receber pacotes legítimos, devido a uma exaustão da memória reservada para a tabela de pacotes de interesses pendentes, ocupada por um excessivo número de interesses falsos. Nesse trabalho, é proposto um modelo de Rede Centrada na Informação que sugere a cooperação entre provedores de conteúdos e roteadores para prover resiliência à prática de IFA. A cooperação facilita a discriminação dos pacotes ilegítimos, fazendo com que o modelo alcance maior eficiência e eficácia nos processos de detecção e mitigação. Resultados obtidos através da simulação de uma rede de cache no backbone da Internet2 evidenciaram a efetividade do modelo proposto.*

1. Introdução

Redes NDN (sigla da expressão em inglês, *Named Data Networks*) [Jacobson et al. 2009] tratam-se de uma das arquiteturas de Redes Centradas na Informação (*Information Centric networks* – ICN) [Xylomenos et al. 2014] que possui, como princípio fundamental, o uso de conteúdos nomeados na comunicação entre produtores e consumidores. O esquema de nomeação de conteúdo adotado em NDNs segue uma organização hierárquica, que é explorada pelos métodos de roteamento e encaminhamento implementados nos roteadores responsáveis pelo gerenciamento dos caches de tais redes [Ioannou and Weber 2016].

Para que seja possível a adoção de um plano de encaminhamento fortemente baseado em nomes de conteúdos, implementações de NDNs fazem uso de pacotes de interesse e de dados. Pacotes de interesse possuem informações sobre quais conteúdos são

requisitados pelos consumidores, enquanto os pacotes de dados transmitem os conteúdos disponibilizados pelos produtores. Os equipamentos de rede utilizam três estruturas de dados para processar e armazenar esses tipos de pacotes: a PIT (*Pending Interest Table*) armazena o nome dos pacotes de interesse que ainda não foram resolvidos e as interfaces que os requisitaram; a FIB (*Forwarding Interest Base*) é utilizada no armazenamento da lista de prefixos dos conteúdos para as interfaces de encaminhamento; e a CS (*Content Store*) realiza o armazenamento temporário de conteúdos disponibilizados na rede. PIT, FIB e CS são estruturas de dados mantidas através de recursos de memória e processamento finitos [Carofiglio et al. 2015, Wang and Hengkui 2015]. Como consequência, tais estruturas são passíveis de serem exploradas em ataques de negação de serviço (do inglês, *Denial of service* – DoS) [AbdAllah et al. 2015].

O IFA (sigla para a expressão em inglês, *Interest Flooding Attack*) é um dos métodos de ataque de DoS em NDNs que tem sido foco de investigação das recentes pesquisas sobre ICN [Mai et al. 2016, AbdAllah et al. 2015]. O método consiste no envio de um excessivo número de pacotes contendo interesses falsos aos roteadores, de forma que os recursos de memória reservados para a PIT sejam esgotados. Este tipo de ataque torna-se viável, dado que interesses não-resolvidos permanecem em memória até alcançar o tempo de expiração. Ao atingir a capacidade total da sua PIT, os roteadores ficam inaptos a processar interesses legítimos, caracterizando, assim, um DoS.

Soluções de detecção e mitigação de DoS com as características do IFA têm sido fortemente investigadas pela comunidade de redes [Mai et al. 2016, Nguyen et al. 2015, Wang et al. 2014, Wang et al. 2013, Compagno et al. 2013, Gasti et al. 2013]. Inicialmente, as soluções de mitigação propostas faziam a mitigação dos efeitos do IFA sem fazer a distinção entre pacotes legítimos e ilegítimos [Compagno et al. 2013, Wang et al. 2014]. Assim, o processo de mitigação implicava em perdas de desempenho da rede para os usuários não maliciosos. Esse problema motivou a criação de métodos mais sofisticados no sentido de tentar mitigar os efeitos do IFA a partir de uma melhor identificação e tratamento de pacotes ilegítimos [Wang et al. 2013, Nguyen et al. 2015]. Atualmente, essa é uma das questões em aberto na área e que ainda carece de investigação. As propostas atuais sugerem que o gerenciamento de pacotes legítimos e ilegítimos seja realizado apenas pelos nós da NDN, o que acarreta numa maior sobrecarga de trabalho no núcleo da rede.

Em contrapartida ao uso dos nós da rede, os provedores de conteúdo possuem um forte potencial para auxiliar no processo de diferenciação entre pacotes legítimos e ilegítimos. Além de possuir mais recursos computacionais, já fazem o gerenciamento dos seus conteúdos localmente, permitindo que o problema seja resolvido de forma distribuída. Por tais motivos, este trabalho apresenta um modelo de NDN em que provedores de conteúdo e nós da rede trabalham de forma cooperativa no processo de detecção e mitigação de IFA. Resultados experimentais obtidos através de simulações do *backbone* da Internet2[1] comprovam a efetividade do modelo. Além do modelo, este trabalho apresenta as seguintes contribuições: i) uma técnica para rápida detecção de DoS em redes de caches; ii) um mecanismo de mitigação de DoS em redes NDN que não sobrecarrega os nós da rede.

[1] www.internet2.edu

Este artigo está organizado da seguinte forma. A Seção 2 discute brevemente os fundamentos e principais características de ataques de negação de serviços do tipo IFA(por interesses falsos) em redes de cache. A Seção 3 apresenta o modelo proposto neste trabalho, detalhando seus elementos e principais características. A Seção 4 descreve o ambiente de avaliação construído para a realização dos experimentos e, a Seção 5, os principais resultados obtidos. Por fim, a Seção 6 apresenta as conclusões do trabalho.

2. Detecção e mitigação de IFA em redes de cache

Em ataques de DoS do tipo IFA, os roteadores ficam inaptos para encaminhar ou receber pacotes de interesses devido à uma exaustão da memória reservada para a PIT [Mai et al. 2016, AbdAllah et al. 2015]. Três abordagens costumam ser empregadas para atingir este objetivo [Gasti et al. 2013]: (i) **uso de conteúdo estático**, que tem o propósito de degradar a eficiência da rede para os consumidores a partir do foco do ataque nos produtores e roteadores que estão associados a conteúdos específicos; (ii) **uso de conteúdo dinâmico**, em que o atacante e seus *zombies* inundam o produtor com pacotes de interesse sobre conteúdos que serão criados dinamicamente pelo produtor; e, por fim, (iii) **uso de interesses falsos**, através do qual, o atacante inunda a PIT do roteador alvo com pacotes de interesse sobre conteúdos inexistentes, ou seja, interesses que jamais serão satisfeitos.

A fim de lidar com tais abordagens, algumas estratégias de mitigação de IFA se baseiam na filtragem ou descarte de pacotes de interesse, assim que os ataques são detectados. Em [Compagno et al. 2013], cada roteador possui uma relação de entrada de pacotes de interesse por saída de pacotes de dados por interface. Quando o valor dessa relação ultrapassa seu limiar, os interesses vão sendo rejeitados. Mensagens de controle são enviadas por essas interfaces cujo limiar foi transposto, para que os roteadores enviem menos pacotes, possibilitando a redução do fluxo de entrada no roteador originário da mensagem. Esse processo se repete até que chegue nos roteadores de borda, impedindo que pacotes maliciosos adentrem a rede. Em [Wang et al. 2014], o ataque é detectado após a taxa de ocupação ou taxa de interesses expirados da PIT ultrapassarem determinados limiares. Então, mensagens de controle percorrem as interfaces de chegada do prefixo apontado como malicioso, até chegarem nos roteadores de borda. Lá, os interesses com o dado prefixo são descartados até que a taxa de recebimento obedeça ao limiar preestabelecido.

Uma característica comum a tais estratégias de mitigação é que não existe distinção entre os pacotes legítimos e ilegítimos no descarte de interesses. Consequentemente, há uma queda de desempenho da rede como um todo durante o processo de mitigação, sobretudo para os usuários não maliciosos.

3. Modelo Proposto

O modelo de NDN proposto neste trabalho baseia-se na colaboração entre produtores e nós da rede para detectar rapidamente o IFA e mitigar seus efeitos sem prejudicar os usuários não maliciosos. A Figura 1 apresenta uma descrição resumida dos principais elementos e suas interações.

A figura mostra que os nós da NDN são responsáveis pela detecção de comportamentos suspeitos. Para confirmar a possível ocorrência de um IFA (1), os nós enviam

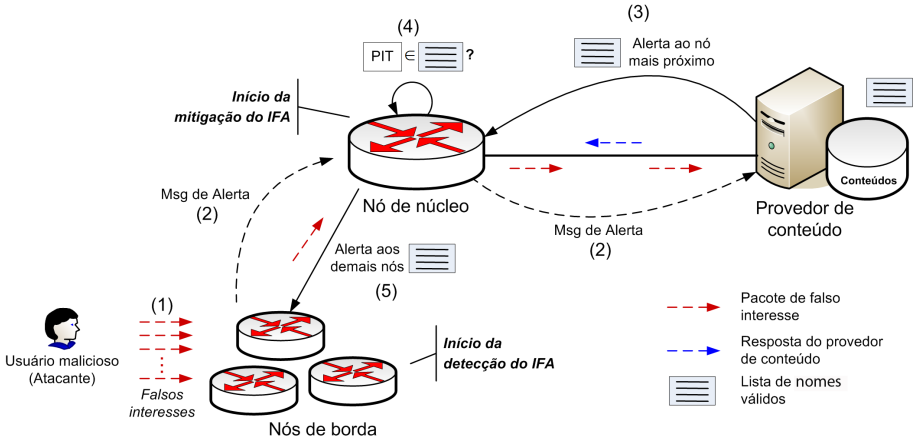


Figura 1. Modelo de NDN proposto em que o nó da rede trabalha de forma colaborativa com os produtores de conteúdo no processo de detecção e mitigação de IFA.

mensagens de alerta aos produtores quando o número de pedidos dos consumidores ultrapassa um determinado limiar (2). Os produtores, que já são responsáveis pelo gerenciamento dos seus conteúdos, enviam aos nós do núcleo uma lista de nomes de dados sob a sua responsabilidade, sempre que uma mensagem de alerta é recebida (3). Ao receber a lista de nomes resolvíveis dos produtores, um nó é capaz de comparar os pedidos da sua PIT e os interesses resolvíveis enviados pelos produtores (4). Em caso de existir inconsistências, o mesmo inicia o processo de mitigação, que consiste em: (i) apagar os interesses falsos de sua PIT; (ii) encaminhar alertas a outros nós envolvidos na comunicação; e (iii) bloquear demais pedidos que estão fora da lista de nomes resolvíveis. Esse mesmo processo de mitigação é propagado para os demais nós da NDN a partir do recebimento da lista de nomes válidos (5). É preciso destacar que a lista de nomes resolvíveis é mantida em memória. Após um determinado período de tempo sem receber interesses falsos ela é removida.

As próximas subseções detalham o papel desses elementos, suas implementações e como as interações ocorrem entre os mesmos, tendo em vista a detecção e mitigação de IFA.

3.1. Estrutura da informação da NDN

A estrutura da informação do modelo proposto se baseia em dois tipos de pacotes: de dados e de interesse. Ambos os tipos tem em sua composição o nome do conteúdo, sendo que o pacote de dados possui também o conteúdo propriamente dito.

Os nomes dos conteúdos estão organizados de forma hierárquica e seguem a seguinte estrutura: prefixo/sufixo. Cada produtor é responsável por um conjunto de dados de um determinado prefixo, cabendo a ele responder aos interesses referentes aos conteúdos armazenados ou que podem ser produzidos sob tal prefixo.

O prefixo identifica o grupo pelo qual o conteúdo pertence. O sufixo destaca unicamente o conteúdo no grupo. Por exemplo, o prefixo "*youtube/*" agrupa todos os vídeos armazenados no produtor que é responsável por ele. O prefixo "youtube/dvdstones" agrupa os conteúdos referentes ao subgrupo "*dvdstones*". A concatenação do prefixo "*youtube/dvdstones*" com o sufixo "parte1" compõe um nome de conteúdo e o identifica

especificamente como membro do subgrupo "*youtube/dvdstones*"

3.2. Produtor de conteúdo

O produtor de conteúdo é o responsável pelo gerenciamento local dos conteúdos disponibilizados na NDN. Seu papel é manter atualizada a lista dos nomes válidos para que os nós da rede possam fazer o processo de mitigação.

Neste trabalho, parte-se da premissa que os produtores participam da NDN conforme um modelo de confiança que garante o envio das listas de nomes resolvíveis sempre que uma mensagem de alerta é recebida. Portanto, assume-se que os produtores não apresentam comportamento malicioso, assim como, não formam algum tipo de conluio com outros produtores no intuito de viabilizar um IFA. Este problema está fora do escopo deste trabalho. Uma discussão mais aprofundada sobre a necessidade de modelos de confiança para prevenir ataques em conluio em cenários de ICN pode ser encontrado em [Salah and Strufe 2016].

3.3. Roteador NDN

Os roteadores desempenham o papel de monitores da rede de forma que o início de um possível IFA possa ser detectado. No modelo proposto, os nós enviam, aos produtores, mensagens de alarme em caso de ocorrência de anomalias no tráfego resultante de um prefixo específico. Além disso, monitoram o comportamento dos produtores. Essa monitoração visa garantir o cumprimento dos protocolos pré-estabelecidos no modelo. Produtores que não seguem o protocolo são descredenciados e todos os prefixos são automaticamente removidos da FIB em questão.

Um roteador NDN possui na sua estrutura os elementos básicos de toda NDN, a saber: PIT, FIB e CS. Complementar a esses elementos, o modelo propõe que cada roteador possua também as seguintes infraestruturas: A tabela de alarmes pendentes (PAT – *Pending Alarm Table*), lista de descartes e o Filtro. Essas estruturas são utilizadas no gerenciamento das mensagens de alarme e na manipulação das listas de nomes válidos enviadas pelos produtores.

A Figura 2 apresenta uma descrição do comportamento das estruturas no roteador NDN. Ao receber um alarme, a PAT o encaminha usando informações da FIB ou descarta caso tenha encaminhado um alarme de mesmo prefixo em um curto intervalo de tempo. Ao receber a resposta ao alarme, é feito o processamento junto com a PIT, que consiste na identificação e descarte dos interesses falsos e no encaminhamento da resposta de alarme para os outros nós da rede que foram percorridos por esses interesses. A CS é utilizada para armazenar conteúdo e resolver pacotes de interesse. Caso o roteador esteja em estado de alerta, o filtro é acionado para identificar se os interesses são resolvíveis, caso não sejam, são descartados.

3.4. Fases do processo de detecção e mitigação

O processo de detecção e mitigação de IFA realizado nos roteadores é composto por três etapas: (i) determinação de limiares, (ii) detecção de anomalias, e (iii) mitigação do IFA.

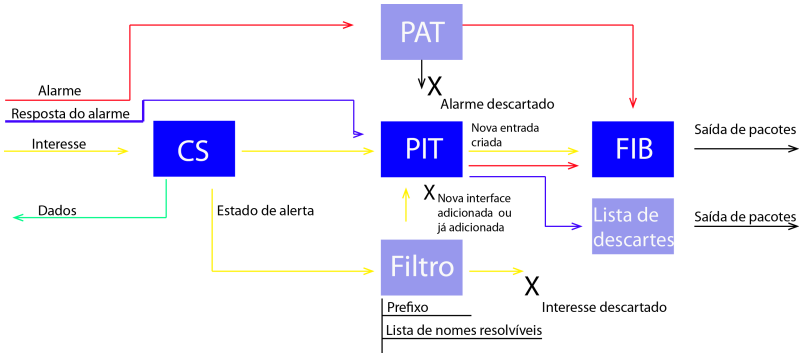


Figura 2. Estrutura de um roteador NDN do modelo proposto.

3.4.1. Determinação de limiares

A determinação de limiares visa estabelecer um limite de entradas para cada prefixo de conteúdos gerenciados pelos produtores. Os roteadores definem os limiares de detecção para cada prefixo armazenado em suas respectivas FIBs. Deste modo, os valores variam para cada roteador e são atualizados dinamicamente, visto que os dispositivos de encaminhamento podem agregar fluxos diferentes e cada prefixo possui um fluxo de tráfego que pode ser diferente dos demais. Seja $P = \{p_1, p_2, \cdots, p_n\}$ a lista de prefixos da NDN, $R = \{r_1, r_2, \cdots, r_m\}$ o conjunto dos roteadores e α_{p_n} a quantidade de pedidos na PIT para o prefixo p_n. l_{p_r} é definido como o limiar l do prefixo p no roteador r. Se, após o envio de um alerta, por conta da ultrapassagem do limiar (ou seja, $\alpha_{p_n} > l_{p_r}$), o IFA for confirmado, o valor de l_{p_r} permanece o mesmo. Caso contrário, trata-se de um falso positivo, então $l_{p_r} \leftarrow \alpha_{p_n}$, estabelecendo assim um novo limiar l_{p_r}. Dado que o padrão dos fluxos da rede pode mudar, cada roteador redefine os limiares de cada prefixo reiniciando a fase de determinação de limiares.

3.4.2. Detecção de anomalias

Após a determinação dos limiares, é realizada a detecção de anomalias, que busca identificar os padrões de requisições que estão fora da normalidade esperada comparando o número de requisições com o limiar estabelecido. Quando o número de requisições excede o limiar, uma mensagem de alerta é enviada ao produtor de conteúdos responsável pelo prefixo.

Após o envio da mensagem de alarme, a PAT é atualizada. A PAT é uma tabela que armazena referências de alarmes recebidos pelo roteador. Cada entrada é composta por prefixo, interface de chegada e o Tempo do Último Encaminhamento (LFT – *Last Forwarding Time*). O roteador que recebe a mensagem de alarme adiciona a interface de chegada do referido prefixo e verifica o LFT. Caso não seja obtida resposta do produtor em um determinado intervalo de tempo, a mensagem de alerta é encaminhada novamente. Esse intervalo entre encaminhamentos é necessário para não inundar a rede. Conforme já mencionado, ao receber uma mensagem de alarme, os produtores enviam uma lista com os nomes de dados que ele é capaz de resolver.

3.4.3. Mitigação

Finalizadas a determinação de limiares e a detecção de anomalias, é feito o processo de mitigação, descrito no Algoritmo 1. A lista enviada pelo produtor é utilizada para identificar a legitimidade dos interesses. O interesse que não constar na lista terá sua interface armazenada em uma lista de descarte e será removido da PIT. As mensagens de resposta percorrem o caminho reverso ao ataque, esse procedimento é essencial para a propagação da mitigação. Após o envio da mensagem de resposta pelo produtor, o roteador entra em estado de alerta, filtrando todos os pacotes que não possam ser resolvidos.

Algorithm 1 Algoritmo de Mitigação

```
1:  procedure MONITORAMENTO
2:  Ao receber um pacote de interesse:
3:      Identifique o prefixo
4:      Atualize a tabela de monitoramento
5:      se Se ocupação estiver acima do limiar então
6:          Mitigação(prefixo)
7:      fim se
8:  procedure MITIGAÇÃO(Prefixo)
9:  Enviar mensagem de alarme para o produtor do prefixo
10: Ao receber uma mensagem de resposta:
11: Para cada entrada na PIT:
12:     se a entrada conter o prefixo então
13:         se a entrada não constar na lista de nomes resolvíveis então
14:             Adicione as interfaces de chegada na lista de descarte
15:             Remova a entrada
16:         fim se
17:     fim se
18: Enviar mensagem de resposta pelas interfaces da lista de descarte
19: Estado de alerta ← true
```

O roteador que receber um interesse não-resolvível durante seu estado de alerta responde com uma mensagem de resposta de alarme pela interface de chegada do pacote. A chegada de um interesse falso em um dispositivo que já ativou a mitigação pode ocorrer quando a resposta a alarmes ainda não alcançou todos os caminhos do ataque. Após enviar mensagens de resposta de alarmes por todas as interfaces maliciosas e estas chegarem aos roteadores de borda, não haverá possibilidade de chegada de interesses falsos, dado que roteadores de borda impedem que os interesses maliciosos adentrem a rede.

Após um tempo determinado sem filtrar pacotes maliciosos, os roteadores saem do estado de alerta. Uma vez que as mensagens resposta de alarme já chegaram a todos os roteadores de borda, que são fontes do ataque, não é mais necessário que as entidades de encaminhamento que estavam no caminho do DoS continuem a filtrar interesses, permitindo, assim, uma redução do uso de recursos computacionais.

Em contrapartida, caso seja constatado que a lista de descarte está vazia, entende-se que todos os pacotes com o prefixo alarmado são resolvíveis. Nesse caso, o incidente é tratado como um falso positivo e, consequentemente, o processo de mitigação não é

disparado. É razoável tal interpretação posto que, se há um ataque em progresso, o dispositivo mais próximo do produtor armazenará e identificará pelo menos um interesse falso, iniciando o processo de mitigação.

É importante salientar que o processo de mitigação ocorre também nos roteadores que não detectaram o ataque ou que estejam em caminhos diferentes do trilhado por uma mensagem de alarme. Isso ocorre pois a mensagens de resposta percorrem caminhos reversos aos fluxos de ataque. Ao receber uma mensagem de resposta, o roteador verifica se contém pacotes maliciosos, podendo assim, disparar a mitigação independente de ter detectado o ataque. Isso propicia um abreviamento dos efeitos da negação de serviço na rede por facilitar a propagação da mitigação.

4. Simulação

Esta seção descreve os cenários, parâmetros, fatores e métricas utilizados no simulador OMNeT++ para avaliação do modelo proposto neste artigo.

4.1. Definição do sistema

O sistema é caracterizado por uma rede NDN formada pelo conjunto C = [0,1,...,n] de consumidores, A = [0,1,...,m] de usuários maliciosos, P = [0,1,...,n] de produtores e R = [0,1,...,j] de roteadores, sendo $r_{borda} \subset$ R e e $r_{bb} \subset$ R, os conjuntos de roteadores de borda e do backbone, respectivamente. Cada roteador possui uma CS e uma PIT. O tamanho da CS e da PIT dos roteadores do *backbone* e de borda são definidos por CS_{bb} e PIT_{bb} e CS_{borda} e PIT_{borda}. Seja D = [0,1,...,n] o conjunto de dados do sistema, cada produtor está associado a um conjunto $D_p \subset$ D, que agrupa dados com um mesmo prefixo. Os consumidores fazem requisições de conteúdos que pertencem a esse conjunto, enquanto os adversários fazem requisições que representam interesses falsos compostos por um prefixo verdadeiro e um sufixo falso, de modo que o interesse chegue a um produtor, porém não possa ser resolvido.

A Figura 3 apresenta a topologia da rede Internet2 utilizada nos experimentos. Foram simulados 13 roteadores no *backbone* (vértices em vermelho), 26 roteadores de borda (vértices em azul) e 8 produtores de conteúdo. Os enlaces de conexão são de 100Gbps entre roteadores e 100 Mbps entre roteador e usuário.

4.2. Cenários, fatores, parâmetros e métricas

O fator avaliado foi a mitigação numa rede que sofre ataques de negação de serviço por inundação de pacotes de interesse. Desse modo, dois cenários associados a esse fator foram analisados: cenário com mitigação e sem mitigação. A avaliação foi realizada de acordo com as seguintes métricas:

- Taxa de ocupação da PIT: avalia o impacto do fluxo malicioso no consumo de memória da PIT. É calculada de acordo com a Equação 1, onde PIT_{req} representa as requisições que estão armazenadas na PIT em um dado instante de tempo t e PIT_{cap} a capacidade da PIT. Essa métrica pode ser medida como uma média geral da ocupação da PIT por cada roteador ou em função do tempo, para verificar o comportamento da taxa de ocupação antes, durante e após o ataque.

$$OcPIT = 100 \times \frac{PIT_{req}}{PIT_{cap}} \quad (1)$$

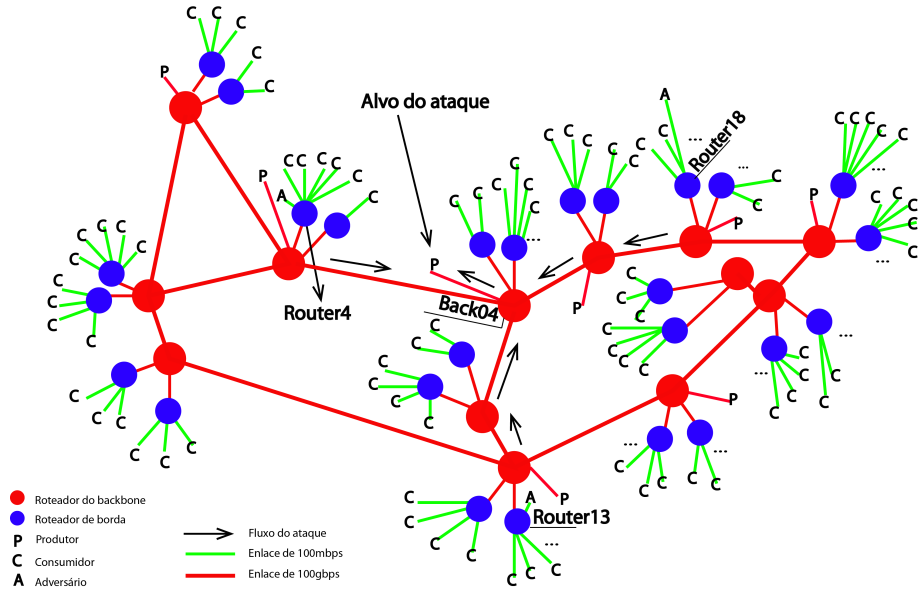


Figura 3. Topologia adaptada do backbone da rede IP da Internet2.

- Taxa de descarte de interesses: essa métrica visa medir o desempenho da rede quanto a satisfação de interesses e pode ser observada na Equação 2, onde I_{desc} refere-se a quantidade de interesses descartados pelo roteador e I_{rec} à quantidade de interesses recebidos. Quanto maior a taxa de descarte, menor é a quantidade de interesses que são resolvidos, indicando portanto, queda no desempenho da rede.

$$DescRot = 100 \times \frac{I_{desc}}{I_{rec}} \tag{2}$$

Os parâmetros utilizados na simulação são definidos na Tabela 1. A rede é composta por 180 nós, sendo 8 produtores, 130 consumidores, 3 adversários, 26 roteadores de borda e 13 do *backbone*. Os 40.000 conteúdos disponíveis foram distribuídos uniformemente entre os produtores e são requisitados a uma taxa de 0-40 pacotes por segundo (pps). Durante o ataque, os adversários solicitam conteúdos falsos a uma taxa de 300-700 pps.

A estimativa do tamanho da PIT foi feita considerando as seguintes premissas: (i) cada consumidor envia rajadas de interesses em intervalos de 0.5s; (ii) - o tempo de resolução de interesses varia de 0.1s a 0.3s e (iii) - A taxa de envio varia de 0 a 20 pacotes por intervalo. Sendo assim, estimou-se o número de entradas suportadas pela PIT em um intervalo de 0.5s, de acordo com a Equação 3. Na qual Router_{maxCap} é a quantidade máxima de nós conectados em um roteador e Taxa_{max} é a taxa máxima de interesses enviados em um intervalo de 0.5s por um consumidor. O resultado desse cálculo é 300. Sendo esta, portanto, a capacidade de armazenamento de entradas da PIT em um roteador de borda. Neste cenário, cada roteador de núcleo agrega dois roteadores de borda, o que acarreta em uma estimativa de 600 entradas. No entanto, foi constatado que 600 entradas era mais do que o suficiente e a fim de economizar memória, a capacidade foi reduzida para 500 entradas neste experimento.

$$PIT_{size} = Router_{maxCap} \times Taxa_{max} \tag{3}$$

Para modelar a taxa de envio de pacotes falsos, foi considerada a capacidade dos roteadores de borda com 300 entradas e o tempo de expiração de 1s. Sendo assim, a taxa mínima do ataque foi de 300 interesses falsos por segundo. Já para garantir a substituição dos interesses existentes por novos interesses falsos, variou-se de 300 a 700 interesses. Com tais valores foi possível assegurar o estouro da PIT e o seu repreenchimento com interesses falsos. Nos experimentos, foram enviados no máximo, 520 mil interesses legítimos (Taxa máxima * Intervalos * nº de consumidores).

Tabela 1. Parâmetros da Simulação

Parâmetro	**Valor**
Quantidade de consumidores	130
Quantidade de adversários	3
Quantidade de roteadores	26 (borda), 13 (*backbone*)
Quantidade de conteúdos disponíveis	40000
Política de descarte de cache	LFU
CSr_{bb}, CSr_{borda}	100
$PITr_{bb}$	500
$PITr_{borda}$	300
Taxa de envio de interesses por consumidores	0 - 40 pps
Taxa de envio de interesses falsos por adversários	300 a 700 pps
Duração da simulação	100s.
Intervalo do ataque	41s - 71s

5. Avaliação dos resultados

Esta seção apresenta a avaliação dos resultados obtidos referente à taxa de ocupação da PIT dos roteadores de borda e de núcleo, assim como o percentual de descarte dos pacotes de interesses.

5.1. Avaliação das taxas de ocupação da PIT

A Figura 4 apresenta a evolução da taxa média de ocupação da PIT dos roteadores de borda "Router4", "Router13" e "Router18". As médias foram calculadas em intervalos de tempo com duração de 1s. Por exemplo, o instante $t = 1$, corresponde à média de valores registrados na PIT no intervalo de 0 a $1s$.

Os roteadores possuem fluxos de tráfego diferentes. Na esquerda estão os resultados obtidos sem o processo de mitigação e na direita com a mitigação. Para fins de comparação, os testes foram executados com os mesmos parâmetros. Como é possível observar, nos testes sem o processo de mitigação, durante o IFA, o percentual de ocupação sobe rapidamente a partir do segundo 41 da simulação, ocupando aproximadamente 90% da memória em todos os roteadores. Já nos resultados apresentados com o processo de mitigação, o mesmo ataque foi reproduzido e, no entanto, a taxa de ocupação não foi afetada, demonstrando a efetividade do processo de mitigação do modelo proposto.

É preciso destacar que os roteadores de borda estão conectados diretamente aos usuários maliciosos. Portanto, agregam um menor fluxo quando comparado com roteadores de núcleo e, assim, são mais sensíveis às variações de tráfego. Deste modo, a anomalia no padrão de consumo da PIT por conta da IFA fica mais evidente.

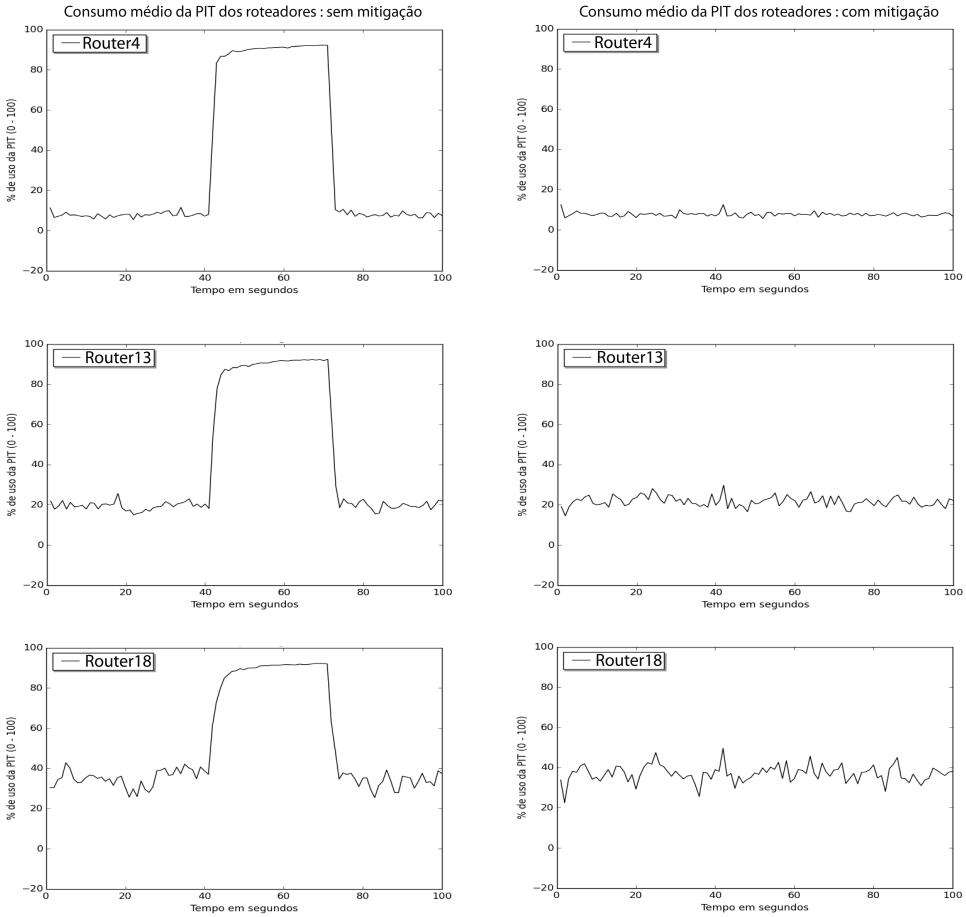


Figura 4. Evolução da taxa de ocupação da PIT dos roteadores de borda com e sem o processo de mitigação.

A Figura 5 apresenta os resultados obtidos na avaliação do roteador de núcleo "Back04" com (a) e sem (b) mitigação. Diferentemente dos de borda, esse roteador está conectado diretamente com o produtor, alvo do IFA. No instante $t = 40$, o processo de determinação de limiares é finalizado e cada roteador fixa os picos como limiares de detecção por prefixo. No instante $t = 41s$, o IFA é iniciado. É possível observar um processo semelhante aos roteadores de borda, exceto que a PIT tem um consumo maior por se tratar de um roteador que agrega tráfego de outros pontos da rede.

Algumas observações precisam ser feitas relacionadas ao experimento. Durante o intervalo do ataque, que é de $t = 41s$ à $t = 71s$, ocorre a saturação da PIT nos roteadores afetados pelo IFA, provocando o descarte de interesses. Entretanto, os valores médios dos intervalos não atingem 100%. Isso é devido ao tempo de expiração dos interesses, que é de $1s$, o que traduz expiração de pacotes em todos os intervalos, exceto pelo intervalo de $t = 0s$ à $t = 1s$. Os valores assumidos em um intervalo de ataque, por exemplo, podem ser 100%, 90%,100% e 88%, resultando em uma média de 94.5%. Ao fim do ataque, os pacotes falsos são expirados e a taxa de ocupação da PIT vai sendo reduzida, dando espaço para a entrada de interesses legítimos. Posteriormente o fluxo volta a sua regularidade.

Diferentemente dos consumidores e usuários maliciosos, os roteadores não enca-

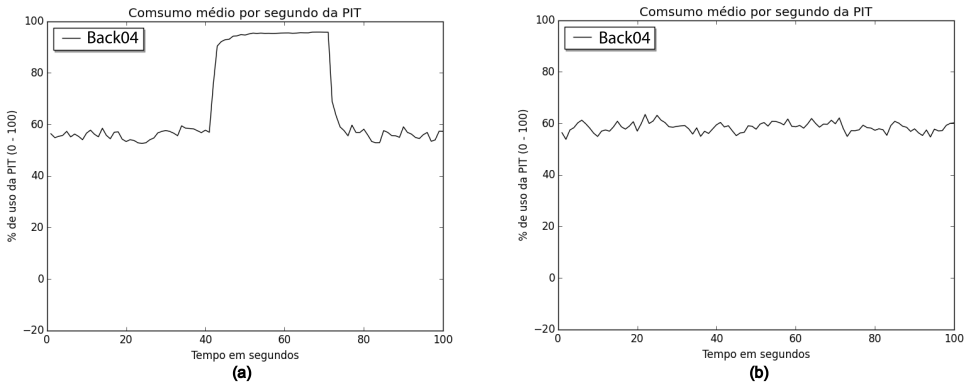


Figura 5. Taxa de ocupação média do roteador de núcleo "Back04". Em (a), sem mitigação. Em (b), com mitigação.

minham as mensagens em rajadas. Assim que a mensagem é processada, ela é encaminhada ou descartada. Nesse contexto, assim que o interesse é adicionado na PIT e constatado a transposição do limiar do seu prefixo, o alarme é enviado em direção ao produtor. A conexão entre roteadores e de roteador com produtor é de 100gbps, enquanto a conexão de roteadores com usuários legítimos/maliciosos é de 100mbps. Essa configuração torna a resposta ao IFA, neste cenário, rápida o bastante para impedir o crescimento da taxa de ocupação média da PIT, como os resultados demonstraram. Após mitigado o ataque, os roteadores de borda filtram os pacotes de interesse, impedindo que os não-resolvíveis adentrem a rede, fazendo com que o tráfego se mantenha regular.

5.2. Avaliação das taxas de descartes

A Figura 6 (a) apresenta a taxa média total de ocupação da PIT dos roteadores de borda e núcleo, com e sem o processo de mitigação do modelo proposto. A Figura 6 (b) apresenta as respectivas taxas de descartes de pacotes de interesses. O percentual com mitigação do "Router4" dos gráficos da Figura 6 (a) é inferior ao percentual sem mitigação. Isso ocorre porque o percentual com mitigação representa o fluxo legítimo recebido por aquele roteador, já que a mitigação descarta a entrada de interesses maliciosos. No entanto, o mesmo não ocorre quando o mecanismo de mitigação é desativado, permitindo a entrada do tráfego malicioso e, consequentemente, aumentando a média total do roteador.

A taxa média total nos roteadores tende a ser maior sem a mitigação devido ao aumento do fluxo provocado pelo ataque. Quanto maior for o fluxo agregado pelo roteador, maior será sua taxa média referente a mitigação. Por exemplo, "Back04" é um roteador de núcleo que agrega regularmente um alto tráfego de pacotes e, por essa razão, recebe o fluxo de ataque oriundo de outros roteadores, aumentando sua taxa média referente a mitigação, e assim reduzindo a variação percentual entre os cenários com e sem mitigação.

Na Figura 6 (b) o percentual de descartes de interesses nos roteadores é calculado através da Equação 2. O percentual com mitigação implica no descarte de pacotes ilegítimos, majoritariamente. Enquanto no cenário sem mitigação, representa o descarte causado pela saturação da PIT. Já a variação do fluxo tem uma importante relevância na taxa de descartes com e sem mitigação. Quando o fluxo malicioso é muito maior do que o legítimo, o roteador tende a descartar mais pacotes pelo processo de mitigação do que

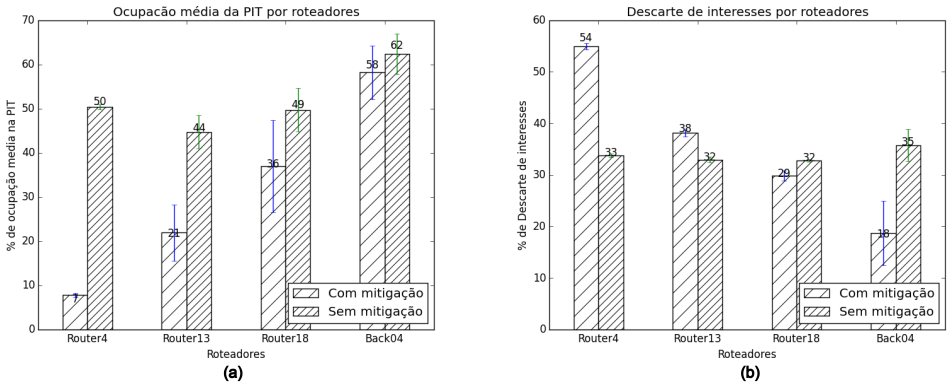


Figura 6. a) Média geral dos valores percentuais assumidos pela PIT dos roteadores. b) Taxa percentual de descartes de interesses por roteadores

por saturação da PIT. Quando o tráfego legítimo é equiparável ao ilegítimo, haverá mais pacotes a serem processados pelo roteador e , por conta disso, terá um aumento na taxa de descarte causado pela insuficiência de memória da PIT do roteador.

Os resultados apresentados para “Router4” e “Router13” demonstram que a taxa com mitigação é maior que a taxa sem mitigação. Nesses roteadores, o fluxo ilegítimo foi maior do que o legítimo. Portanto, o descarte pela mitigação é maior do que por saturação. Já os roteadores “Router18” e “Back04”, que agregam um maior fluxo na rede, tendem a descartar mais pacotes por saturação do que por mitigação. Por essa razão, as taxas sem mitigação são maiores.

Na Figura 6 (b), o percentual com mitigação no roteador “Back04” é menor em comparação aos outros roteadores. Isso ocorre devido a PIT dos roteadores de núcleo serem maiores do que a PIT dos roteadores de borda. Sendo assim, apresenta percentuais relativamente menores. Uma outra implicação dessa característica é que os roteadores de borda atuam como gargalos do ataque, descartando uma grande quantidade do tráfego malicioso que recebe, reduzindo assim os efeitos do IFA nos roteadores de núcleo.

6. Conclusão e Trabalhos Futuros

De acordo com os resultados apresentados na Seção 5, o modelo proposto se mostra eficaz na mitigação de ataques de IFA por interesses falsos. Isso acontece devido a sua capacidade de mitigar o ataque sem descartar interesses legítimos preservando o desempenho da rede, a propagação da mitigação pelos roteadores e o descarte de novos interesses maliciosos. Para trabalhos futuros, será realizada a simulação em redes com maior número de nós, sobretudo roteadores, para prover a validação da escalabilidade do modelo. Além disso, o modelo deverá comportar uma proposta de modelo de confiança de produtores para que a rede possa mitigar o ataque, mesmo com a inclusão de produtores em conluio com atacantes.

7. Agradecimentos

Os autores agradecem o apoio da FAPESB e CAPES.

Referências

AbdAllah, E. G., Hassanein, H. S., and Zulkernine, M. (2015). A survey of security attacks in information-centric networking. *IEEE Communications Surveys Tutorials*, 17(3):1441–1454.

Carofiglio, G., Gallo, M., Muscariello, L., and Perino, D. (2015). Pending interest table sizing in named data networking. In *Proceedings of the 2Nd ACM Conference on Information-Centric Networking*, ACM-ICN '15, pages 49–58, New York, NY, USA. ACM.

Compagno, A., Conti, M., Gasti, P., and Tsudik, G. (2013). Poseidon: Mitigating interest flooding ddos attacks in named data networking. In *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*, pages 630–638. IEEE.

Gasti, P., Tsudik, G., Uzun, E., and Zhang, L. (2013). Dos and ddos in named data networking. In *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*, pages 1–7. IEEE.

Ioannou, A. and Weber, S. (2016). A survey of caching policies and forwarding mechanisms in information-centric networking. *IEEE Communications Surveys Tutorials*, PP(99):1–1.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009). Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM.

Mai, H. L., Nguyen, N. T., Doyen, G., Ploix, A., and Cogranne, R. (2016). *On the Readiness of NDN for a Secure Deployment: The Case of Pending Interest Table*, pages 98–110. Springer International Publishing, Cham.

Nguyen, T., Cogranne, R., and Doyen, G. (2015). An optimal statistical test for robust detection against interest flooding attacks in ccn. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 252–260.

Salah, H. and Strufe, T. (2016). Evaluating and mitigating a collusive version of the interest flooding attack in ndn. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 938–945.

Wang, K. and Hengkui, W. (2015). Tcam-pc: Space-efficient tcam-based packet classification with packet-forwarding-rate constraints. In *2015 12th IEEE International Conference on Electronic Measurement Instruments (ICEMI)*, volume 01, pages 260–264.

Wang, K., Zhou, H., Qin, Y., Chen, J., and Zhang, H. (2013). Decoupling malicious interests from pending interest table to mitigate interest flooding attacks. In *2013 IEEE Globecom Workshops (GC Wkshps)*, pages 963–968.

Wang, K., Zhou, H., Qin, Y., and Zhang, H. (2014). Cooperative-filter: countering interest flooding attacks in named data networking. *Soft Computing*, 18(9):1803–1813.

Xylomenos, G., Ververidis, C. N., Siris, V. A., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, K. V., and Polyzos, G. C. (2014). A survey of information-centric networking research. *Communications Surveys & Tutorials, IEEE*, 16(2):1024–1049.

Trilha Principal do SBRC 2017
Sessão Técnica 23
Redes Definidas por Software III

Alocação de Infraestruturas Virtuais em *data centers* implementados com Redes Definidas por *Software*

Felipe Rodrigo de Souza, Charles C. Miers, Adriano Fiorese, Guilherme Koslovski

[1]Programa de Pós-Graduação em Computação Aplicada
Universidade Estadual de Santa Catarina (UDESC) – Joinville/SC – Brasil
dcc6frs@joinville.udesc.br, {*charles.miers, adriano.fiorese, guilherme.koslovski*}*@udesc.br*

Resumo. *A virtualização de data centers combinada com o paradigma de Software-Defined Networking (SDN) impulsionaram a criação de serviços nas nuvens computacionais. Dentre as atividades gerenciais de um provedor de nuvem baseado em SDN, o presente trabalho aborda a alocação de infraestruturas virtuais com requisitos de Quality-of-Service (QoS). Inicialmente, o problema é modelado como programação inteira mista. Depois, restrições em variáveis são relaxadas e uma heurística para aproximação dos resultados é discutida. Além de alocar a qualidade de serviço solicitada, a análise experimental indica a redução da latência de comunicação observada por clientes.*

Abstract. *The virtualization of data centers combined with the SDN paradigm lead to the provisioning of new cloud services. Among the management challenges faced by a SDN-based cloud provider, this work improves the virtual infrastructure allocation problem with QoS requirements. First, the problem is modeled as a mixed integer programing. Thus, integer constraints are relaxed and rounding heuristics are discussed. Besides guaranteeing the QoS requirements, the results indicate a reduction on virtual network communication latency.*

1. Introdução

Com a computação em nuvem, os clientes tem acesso a recursos computacionais fornecidos pelos provedores, de maneira dinâmica e de acordo com a demanda de suas aplicações [Mell and Grance 2011]. Provedores de *Infrastructure-as-a-Service (IaaS)* tem utilizado virtualização para provisionar Infraestruturas Virtuais (IVs), que são compostas por Máquinas Virtuais (MVs) em conjunto com serviços de *Network-as-a-Service (NaaS)* [Fischer et al. 2013, Manvi and Shyam 2014]. Dentre os desafios enfrentados por provedores IaaS, a alocação de IVs consiste em identificar dentro do *data center* um conjunto de recursos para hospedar os componentes da IV. A alocação de IVs pode ser vista como um mapeamento de grafos. Os vértices representam os equipamentos (processamento e encaminhamento) e as arestas representam os enlaces de comunicação. Assim, o problema consiste em mapear o grafo que representa a IV no grafo que representa o *data center*, caracterizado como um problema NP-difícil [Fischer et al. 2013].

A complexidade do processo de alocação tende a aumentar, visto que são criados cada vez mais critérios para garantir a qualidade da IV provisionada. Por exemplo, aplicações em IVs geram grandes volumes de tráfego e boa parte do tempo de execução é devido a transferência de dados. Um *cluster* do Facebook pode consumir até 33% de seu tempo de execução somente realizando transferências [Rost et al. 2015]. É crescente

o número de aplicações na nuvem que solicitam requisitos de rede, sendo um mecanismo de alocação focado somente em MVs uma visão simplista.

Recentemente, *Software-Defined Networking (SDN)* surgiu como uma alternativa para simplificar o provisionamento de IVs. SDN separa a parte de controle, normalmente embutida nos equipamentos de rede, para um controlador externo e logicamente centralizado. Além disso, fornece mecanismos para isolamento de IVs e garantia de *Quality-of-Service (QoS)* [Sherwood et al. 2009]. Apesar de SDN facilitar tanto o processo de provisionamento de IVs quanto a representação dos recursos de encaminhamento, algumas características exclusivas do ambiente introduzem outros níveis de complexidade. Em SDN, a garantia de largura de banda é realizada diretamente nos *switches*. Portanto, todos os equipamentos para alocação de um enlace virtual devem conter regras para respeitar os limites preestabelecidos. Ainda, as entradas nas tabelas de encaminhamento dos *switches* são limitadas [Kreutz et al. 2015], o que leva ao armazenamento temporário de entradas no controlador. Assim, a eventual latência até o controlador deve ser considerada no cálculo para verificação da conformidade com os requisitos de QoS do cliente.

A contribuição do presente trabalho é a proposta de um mecanismo intitulado QVIA-SDN (*Quality-Aware Virtual Infrastructure Allocation SDN*), que é composto por um modelo para alocação de IVs em um *data center* de nuvem baseado em SDN, com *switches* OpenFlow. O problema é modelado como programação inteira mista, depois as restrições em variáveis são relaxadas e uma heurística para aproximação dos resultados é discutida. Os resultados experimentais indicam que além de respeitar o QoS, há uma redução na latência média percebida por clientes de IVs.

Este trabalho está organizado da seguinte forma. A Seção 2 contextualiza a proposta frente aos trabalhos relacionados. A Seção 3 apresenta a formulação do problema, enquanto a Seção 4 descreve um *Mixed Integer Program (MIP)* ótimo para alocação de IVs. O relaxamento das variáveis e as heurísticas são abordados na Seção 5. A análise experimental é discutida na Seção 6 e as considerações finais são apresentadas na Seção 7.

2. Trabalhos relacionados

A literatura define que o provisionamento de IVs em *data centers* de *IaaS* deve ser guiado por políticas, que podem ser aplicadas com foco local (por enlace físico ou caminho congestionado) ou global (considerando a topologia do *data center*) [Ballani et al. 2011]. No trabalho de [Popa et al. 2011] foi proposto o compartilhamento proporcional dos recursos de rede, e isso só pode ser alcançado em um cenário controlado, como SDN, visto que requer coordenação entre o controlador e os recursos virtuais. As políticas são baseadas em parâmetros que configuram o compartilhamento dos caminhos físicos que estão alocando enlaces virtuais, alinhado com a nossa proposta: parâmetros indicados pelos clientes (largura de banda e latência) definem os requisitos para configuração dos enlaces.

Quanto a seleção de recursos físicos para alocar IVs, é possível encontrar propostas de formulação ótima e heurísticas de aproximação [de Oliveira and Koslovski 2017, Cavalcanti et al. 2014, Fischer et al. 2013]. Em geral, cada proposta é guiada por um objetivo diferente, com destaque para [Chowdhury et al. 2012], que inovou ao propor a alocação conjunta de recursos de processamento e comunicação (discutido na Seção 4).

A alocação de IVs em *data center* SDN segue um desafio para provedores de nuvem IaaS. [Sherwood et al. 2009] identificou os principais desafios na alocação de re-

cursos SDN, destacando as tabelas de fluxo e o compartilhamento de controladores entre as IVs alocadas. Além disso, propôs um mecanismo para provisionar redes virtuais semelhante aos *hypervisores* de MVs. De maneira complementar, [Al-Shabibi et al. 2014] propôs um *framework* para virtualização de redes utilizando SDN. O presente trabalho propõe um algoritmo de alocação baseado em QoS (Seção 5) que considera os desafios e particularidades do processo de alocação [Sherwood et al. 2009, Al-Shabibi et al. 2014].

[Demirci and Ammar 2014] identificou que o posicionamento do controlador tem impacto no desempenho da aplicação, devido a latência até o controlador. Propostas de garantia de largura de banda em ambientes SDN estão presentes em [Tao et al. 2015, Mijumbi et al. 2014]. No presente trabalho, largura de banda e latência são impostas através da formulação MIP relativa ao ambiente SDN. Ainda, o MIP proposto na Seção 4 considera que os *switches* podem ser reservados e gerenciados pelos clientes.

É importante ressaltar que o presente trabalho inova na alocação de IVs com requisitos de QoS em um *data center* de nuvem baseado em SDN, considerando os desafios indicados na literatura. Além disso, o problema de alocação é formulado como uma alocação conjunta de MVs, *switches* e enlaces, alegando que o desempenho das aplicações hospedadas na nuvem pode ser impactado por políticas de alocação de rede.

3. Formulação do problema

3.1. *Data center* IaaS e requisições de IVs

Tanto a infraestrutura de *data center* quanto as requisições de IVs são representadas por grafos não direcionados com pesos. O *data center* é representado pelo grafo $G^s(N_h^s, N_n^s, C^s, E^s)$, sendo que N_h^s representa o conjunto de servidores e N_n^s o conjunto de *switches* OpenFlow que compõem a infraestrutura física. A notação utilizada ao longo do trabalho é resumida na Tabela 1.

Notação	Descrição	Notação	Descrição
$G^s(N_h^s, N_n^s, C^s, E^s)$	grafo do *data center*	$G^v(N_h^v, N_n^v, E^v, D^v)$	Requisição de IV
N_h^s	servidores físicos	N_h^v	MVs
N_n^s	*swiches* SDN	N_n^v	*switches* virtuais
C^s	controlador SDN	E^v	enlaces virtuais
E^s	enlaces físicos	D^v	matriz de latência máxima aceita
$R(.)$	capacidade física residual	d_{ij}	latência máxima entre i e j
$\Omega(i)$	candidatos físicos para hospedar i	c_i	capacidade solicitada para i
$P^s(i,j)$	caminhos entre $\Omega(i)$ e $\Omega(j)$		

Tabela 1. Notação utilizada ao longo do artigo. i e j representam recursos virtuais, enquanto u e v são utilizados para recursos físicos.

Os controladores SDN são representados por C^s. Os enlaces que interconectam os servidores e equipamentos de rede são representados pelo conjunto E^s. Cada recurso (servidores, equipamentos de rede ou enlaces) possui uma capacidade residual, denotada pela função $R(.)$. De maneira similar, uma requisição de IV é um grafo $G^v(N_h^v, N_n^v, E^v, D^v)$, sendo que N_h^v representa o conjunto de MVs, N_n^v o conjunto de equipamentos de rede e

E^v o conjunto de enlaces. D^v é uma matriz que contêm a latência máxima aceita fim-a-fim, na qual d_{ij} representa a maior latência aceita entre os recursos i e j. A capacidade solicitada para cada recurso virtual é representada por c. Como a configuração da rede virtual é um aspecto importante para o desempenho de aplicações hospedadas em IVs, provedores estão buscando a alocação de IVs com requisitos de QoS de rede. Neste sentido uma requisição de IV pode ser baseada em IaaS e NaaS, exemplificado na Figura 1.

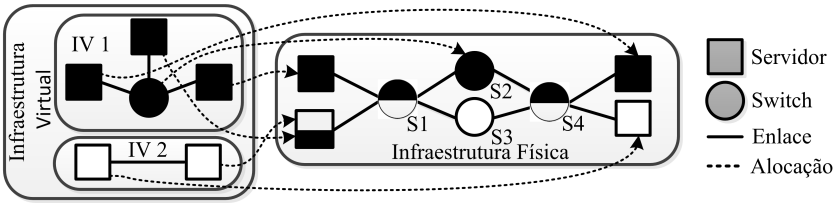


Figura 1. Alocação de IVs em provedores de nuvens baseados em SDN.

Requisições somente IaaS representam MVs solicitadas sem a descrição da topologia virtual. Entretanto, além das configurações de MVs, regiões ou zonas, alguns requisitos de rede, como por exemplo a latência máxima fim-a-fim ou uma largura de banda mínima podem ser especificados. Este tipo de requisição é representado pela IV 2 (Figura 1), na qual duas MVs devem ser provisionadas com requisitos de QoS de rede. É importante destacar que os equipamentos de rede utilizados para garantir a comunicação das MVs são abstraídos do cliente sendo de conhecimento somente do provedor.

Requisições combinando IaaS e NaaS permitem a completa descrição da IV. O cliente pode realizar uma descrição detalhada dos equipamentos de rede que compõem a rede virtual, bem como os requisitos de QoS. A IV 1 (Figura 1) exemplifica que o provedor pode utilizar mais de um recurso físico para para alocar um virtual: enquanto a IV solicitou um único *switch* virtual, o provedor utilizou 3 *switches* SDN físicos para garantir a comunicação dos equipamentos virtuais. Neste caso, o cliente só tem acesso para configuração do *switch* que efetivamente está alocando o equipamento virtual solicitado.

3.2. Alocando recursos físicos para hospedar IVs

As requisições de IVs são processadas individualmente pelo *framework* de alocação da nuvem, o qual é responsável por tomar a decisão de aceitar ou não uma IV. Os requisitos da IV podem alterar durante seu ciclo de vida, entretanto este trabalho não aborda a reconfiguração, pois acreditamos que alocação e reconfiguração devem ser analisadas individualmente. Ao fim da execução da IV, os recursos destinados a mesma são liberados.

A alocação de IVs em um *data center* de nuvem pode ser dividida em duas etapas: alocação de nós e alocação de enlaces [Chowdhury et al. 2012]. O mapeamento de MVs em servidores físicos é dada por $M_h : N_h^v \mapsto N_h^s$, enquanto para *switches* virtuais mapeados em *switches* SDN por $M_n : N_n^v \mapsto N_n^s$, sendo $M_h(i) \in N_h^s$ e $M_n(i) \in N_n^s$. De maneira similar, o mapeamento de um enlace virtual ij é realizado em um caminho $p \in P^s$, entre os nós que alocam os nós virtuais finais (*switch* ou MV) do enlace ij. Para hospedar um recurso virtual, o nó físico deve ter uma capacidade residual (não reservada) maior ou igual a capacidade virtual solicitada. Para as MVs e os *switches*, $c_i \leq R(M_h(i))$ e $c_j \leq R(M_n(j))$, respectivamente, enquanto para os enlaces virtuais, $c_{ij} \leq R(M_e(ij))$.

3.3. Objetivos do provedor de nuvem IaaS

O principal objetivo neste trabalho é a alocação de IVs com requisito de QoS de rede. Além disso, a formulação busca a redução do custo de alocação, aumentando o *revenue* do provedor. De maneira similar a outros trabalhos, o custo para alocar uma IV é proporcional a capacidade física reservada para garantir o *Service Level Agreement (SLA)*, como mostrado na Eq. (1), onde $|ij|$ representa o tamanho do caminho para alocar ij, em número de saltos [Fischer et al. 2013]. O *revenue* ao alocar uma IV é dado pela Eq. (2), visto como o somatório das capacidades requisitadas.

$$\mathcal{C}(G^v) = \sum_{i \in N_h^v} c_i + \sum_{j \in N_n^v} c_j + \sum_{ij \in E^v} c_{ij}|ij| \tag{1}$$

$$\mathcal{R}(G^v) = \sum_{i \in N_h^v} c_i + \sum_{j \in N_n^v} c_{ij} + \sum_{ij \in E^v} c_{ij} \tag{2}$$

A *Quality-of-Service (QoS)* fornecida ao cliente é medida pela latência média dos recursos virtuais comunicantes, como descrito na Eq. (3), na qual $\mathcal{L}(i,j)$ representa a latência média do caminho físico alocando ij, e $|E^v|$ indica o número de enlaces virtuais.

$$\mathcal{Q}(G^v) = \frac{\sum_{ij \in E^v} \mathcal{L}(i,j)}{|E^v|} \tag{3}$$

4. MIP ótimo para alocação de IVs com requisitos de QoS

4.1. Seleção de candidatos para alocar recursos virtuais

Em provedores públicos de IaaS, clientes podem selecionar regiões e zonas para alocação das MVs. Assim, cada recurso virtual requisita uma localização (zona ou região) representado por $loc(i)$. Qualquer recurso físico na localização $loc(i)$ é um candidato para alocar i. Candidatos para alocar o recurso virtual i são representados pelo conjunto $\Omega(i)$.

Além da localização, requisitos de QoS de rede são essenciais para inquilinos [Stallings 2015]. Sobretudo, a latência e o desempenho da rede virtual não são fatores totalmente dependentes da localização física [Persico et al. 2015]. Sendo assim, além da seleção usual baseada na localização [Mijumbi et al. 2015] [Chowdhury et al. 2012], é proposta a seleção de candidatos baseada em latência.

Candidatos para alocar *switches* virtuais com requisitos de latência. Para requisições de IVs com enlaces contendo requisitos de latência e *switches* virtuais, o conjunto de candidatos físicos para alocar o *switch* i é composto por equipamentos que possuem enlaces capazes de hospedar o pior requisito de latência de i. Em suma, $\Omega(i) = \{u \in N_n^s | max(lat(u,v)) < max(d_{ij})\}; \forall v \in adj(u); \forall j \in adj(i)$, onde $adj(.)$ retorna o conjunto de adjacentes, e $lat(u,v)$ indica a latência do caminho físico entre u e v.

Candidatos para alocar *switches* virtuais sem requisitos de latência. Neste caso, todos os *switches* físicos com capacidade residual suficiente são candidatos para alocar o *switch* virtual i. Assim, $\Omega(i) = \{u \in N_n^s | R(u) \geq c_i\}$.

Candidatos para alocar MVs sem requisitos de latência. Os servidores físicos com suficiente capacidade residual são selecionados para alocar a MV i. Em outras palavras, $\Omega(i) = \{u \in N_h^s | R(u) \geq c_i\}$.

Candidatos para alocar MVs com requisitos de latência.
(a) MVs conectadas com *switches* virtuais: neste caso, os candidatos são selecionados com base na latência de comunicação do *switch* conectado com MV i. Assim, $\Omega(i)$ é composto por $\{u \in N_h^s | lat(u,v) \leq d_{ij}\}; \forall v \in \Omega(j); j \in adj(i) \setminus N_h^v$.
(b) MVs conectadas com MVs: os candidatos, são selecionados com base nos requisitos de latência fim-a-fim. Portanto, $\Omega(i) = \{u \in N_h^s | lat(u,v) \leq d_{ij}\} \forall v \in N_h^s$.

Seguindo a abordagem proposta por [Chowdhury et al. 2012], cada recurso virtual i é inserido no grafo físico, através de um enlace temporário com seus candidatos, com capacidade infinita e sem latência. O grafo aumentado resultante é representado por $G^{s'}(N^{s'}, C^s, E^{s'})$. Assim, $N^{s'} = N_h^s \cup N_h^v \cup N_n^s \cup N_n^v$; e $E^{s'} = E^s \cup \{iu \,|\, i \in N_h^v, u \in \Omega(i)\} \cup \{ju \,|\, j \in N_n^v, u \in \Omega(j)\}$. Os controladores físicos (C^s) não são afetados pela criação do grafo aumentado. A Figura 2 exemplifica um grafo aumentado, conectando a IV 1 da Figura 1 a um *data center* simplificado.

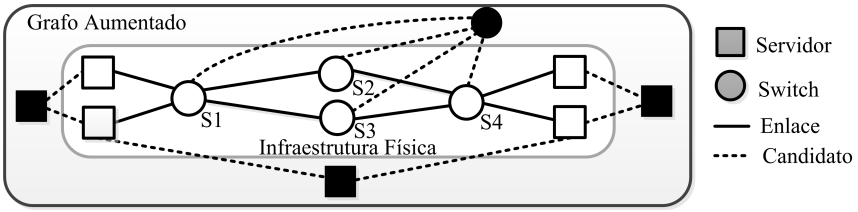


Figura 2. Grafo aumentado combinando recursos físicos e virtuais.

4.2. Variáveis e objetivo

Para representar uma solução de mapeamento de IVs com QoS de rede foram utilizadas três variáveis. A variável f_{ijuv} representa o quanto do fluxo (largura de banda solicitada para o enlace virtual) de ij está alocado no enlace físico entre u e v. Com domínio binário, a variável x_{uv} representa a presença ou não de um fluxo (requisição de enlace virtual) entre os recursos u e v. Se a condição $\sum_{ij \in E^v}(f_{ijuv} + f_{ijvu}) > 0$ for atendida o valor em x_{uv} é definido como 1, caso contrário 0. Por fim, a variável binária e_{ijpu} representa as entradas das tabelas de encaminhamento de fluxos que estão alocadas no controlador (1 em caso de ocorrência). Quando 0, o fluxo ij está alocado sobre o caminho p no *switch* u.

De acordo com os objetivos do provedor (Seção 3.3), uma versão modificada das Equações (1), (2) e (3) compõem a função objetivo (Eq. 4). A minimização da função objetivo reduz o custo para alocação de uma IV, através da redução do número de servidores e enlaces físicos utilizados, bem como pela alocação de fluxos no controlador. Como proposto na literatura, ao ponderar o custo pela capacidade residual dos recursos físicos (servidores, *switches* e enlaces), são selecionados os recursos com maior capacidade residual, balanceando a carga do *data center* [Chowdhury et al. 2012]. Os parâmetros α_{uv}, β_u, e γ_e controlam a importância de cada recurso de acordo com a visão do provedor, com valores entre 1 e $R(.)$, enquanto δ é um número positivo para evitar divisão por zero.

$$min: \sum_{u \in N_h^s \cup N_n^s} \frac{\beta_u}{R(u)+\delta} \sum_{i \in N^v} x_{iu} c_i + \sum_{u \in N_n^s} \frac{\gamma_e}{R(u)+\delta} \sum_{ij \in E^v} \sum_{p \in P^s(i,j)} (1 - e_{ijpu}) + \sum_{uv \in E^s} \frac{\alpha_{uv}}{R(uv)+\delta} \sum_{ij \in E^v} f_{ijuv} \quad (4)$$

4.3. Restrições

Para garantia dos requisitos de QoS solicitados no SLA, um conjunto de restrições de capacidade, fluxo, binárias e meta restrições deve ser atendido pelo mecanismo de alocação.

Restrições de capacidade dos servidores e enlaces. A Eq. (5) define a restrição de capacidade para os enlaces, enquanto a Eq. (6) se aplica aos servidores. Em suma, os recursos físicos para devem ter capacidade residual suficiente para alocar as requisições.

$$\sum_{ij \in E^v} (f_{ijuv} + f_{ijvu}) \leq R(uv) x_{uv} \qquad \forall u, v \in N^{s'} \tag{5}$$

$$R(u) \geq \sum_{i \in N_h^v} x_{iu} c_i \qquad \forall u \in N_h^s \tag{6}$$

Restrições de SDN. Os equipamentos com SDN possuem uma particularidade quanto as suas tabelas de encaminhamento: os fluxos que passam por um determinado *switch* (representado por x) podem estar alocados somente nas tabelas de encaminhamento do controlador (identificado por e). Neste sentido, a Eq. (7) busca garantir que a capacidade residual de um *switch* u deve ser suficiente para alocar todos os fluxos que passam por aquele *switch*, entretanto, as entradas alocadas no controlador não devem ser consideradas. s_{ij} e t_{ij} representam a origem e o destino, do enlace virtual ij.

$$R(u) \geq \sum_{k \in N_n^v} \sum_{ij \in E^v : s_{ij} = k \vee t_{ij} = k} \sum_{p \in P^s(i,j)} (x_{ku} - e_{ijpu}) c_k + \sum_{ij \in E^v} \sum_{p \in P^s(i,j)} (x_{iu} - e_{ijpu}) \qquad \forall u \in N_n^s \tag{7}$$

Restrições de fluxo de dados. As requisições de enlaces virtuais, representam um fluxo que deve ser transferido no *data center*. Assim, a Eq. (8) garante que para cada enlace virtual ij, o fluxo iniciado em s_{ij} devem ser igual a capacidade solicitada para o enlace virtual, enquanto a Eq. (9) tem significado equivalente para o fluxo chegando em t_{ij}. Complementando, a Eq. (10) é responsável pelos fluxos encaminhados no *data center* SDN: cada *switch* intermediário deve encaminhar os fluxos que recebeu em sua totalidade.

$$\sum_{u \in N^{s'}} f_{ijs_{ij}u} - \sum_{u \in N^{s'}} f_{ijus_{ij}} = c_{ij} \qquad \forall ij \in E^v \tag{8}$$

$$\sum_{u \in N^{s'}} f_{ijt_{ij}u} - \sum_{u \in N^{s'}} f_{ijut_{ij}} = -c_{ij} \qquad \forall ij \in E^v \tag{9}$$

$$\sum_{v \in N^{s'}} f_{ijuv} - \sum_{v \in N^{s'}} f_{ijvu} = 0 \qquad \forall ij \in E^v, \forall u \in N^{s'} \setminus \{s_{ij}, t_{ij}\} \tag{10}$$

Restrições de latência. As Equações (11) e (12) garantem que a latência do caminho físico alocando o enlace virtual é menor ou igual ao solicitado, mesmo quando algumas regras de encaminhamento estão no controlador.

$$d_{ij} \leq \sum_{u,v \in p} (lat(u,v) x_{uv} + lat(u,c) e_{ijpu}) \forall ij \in E^v; \forall p \in P^s(i,j) \tag{11}$$

$$d_{ij} \geq \sum_{u,v \in p} lat(u,v) x_{uv} \forall ij \in E^v; \forall p \in P^s(i,j) \tag{12}$$

Meta restrições e restrições binárias. Pela Eq. (13), um recurso virtual será alocado em somente um recurso físico, enquanto a Eq. (14) garante que x irá receber valores quando existir algum fluxo passante. Eqs. de (15) a (17) definem os domínios das variáveis.

$$\sum_{u \in \Omega(i)} x_{iu} = 1 \qquad \forall i \in N^v \tag{13}$$

$$x_{uv} = x_{vu} \qquad \forall u, v \in N^{s'} \tag{14}$$

$$f_{ijuv} \geq 0 \qquad \forall u, v \in N^{s'}; \forall ij \in E^v \tag{15}$$

$$x_{uv} \in \{0, 1\} \qquad \forall u, v \in N^{s'} \tag{16}$$

$$e_{ijpu} \in \{0, 1\} \qquad \forall u \in N_n^s; \forall ij \in E^v; \forall p \in P^s(i, j) \tag{17}$$

5. QVIA-SDN

É fato que a solução de um MIP é um problema NP-difícil [Chowdhury et al. 2012]. Diante da limitação da aplicabilidade da formulação ótima, as restrições binárias são relaxadas, compondo um *Linear Program (LP)* Em um segundo momento, o número de candidatos e caminhos físicos são reduzidos. Após a solução do LP, os resultados são tratados por uma heurística para obtenção de uma solução. QVIA-SDN (*QoS-Aware VI Allocation on SDN-based data-centers*) é composto pelas técnicas listadas.

5.1. Relaxando as variáveis

Para obter o LP, as Eqs. (16) e (17) tiveram seus domínios relaxados. Assim, respectivamente, $1 \geq x_{uv} \geq 0; \forall u, v \in N^{s'}$ e $1 \geq e_{ijpu} \geq 0; \forall u \in N_n^s; \forall ij \in E^v; \forall p \in P^s(i, j)$. A função objetivo bem como as demais restrições permanecem inalteradas no LP.

5.2. Redução do número de candidatos físicos

Os *data centes* normalmente são compostos por recursos homogêneos, interconectados por uma infraestrutura de rede estruturada [Stallings 2015]. Assim, recursos podem ser agrupados de acordo com aspectos comuns (*e.g.*, largura de banda, latência, CPU, RAM). Esta informação pode ser utilizada para composição de grupos de candidatos [Rost et al. 2015]. Nesta linha, QVIA-SDN reduz o número de candidatos físicos utilizando somente uma porcentagem de cada região do *data center*. Esta informação pode ser ajustada pelo provedor de acordo com a quantidade de recursos físicos disponíveis por região ou zona. É esperado que ao reduzir o número de candidatos, a eficiência do algoritmo seja impactada. Essa parametrização é discutida na Seção 6.

É comum que topologias de *data center* tenham caminhos redundantes [Stallings 2015, Al-Fares et al. 2008], gerando um grande volume de caminhos, que são impraticáveis de calcular a cada alocação. A contabilização de todos os caminhos entre os recursos que compõem um *data center*, a cada alocação, é impraticável. Como o LP requer um conjunto P^s para encontrar uma solução aproximada, QVIA-SDN emprega uma redução de caminhos físicos candidatos. Essa redução somente oculta a existência de caminhos, permitindo futuramente a aplicação de algoritmo de resiliência ou engenharia de tráfego. Assim, P^s é composto por um (i) caminho mais curto e (ii) um caminho com a menor latência, entre todos os candidatos físicos (servidores e *switches*), sendo (i) diferente de (ii).

5.3. Heurísticas

Ao resolver o LP relaxado, a correlação entre as variáveis f, x e e é perdida. Portanto, QVIA-SDN aplica uma heurística para avaliar os valores obtidos e aproximar a solução. A heurística é composta por duas etapas, como descrito pelos Algoritmos 1 e 2.

Input: G^v, G^s
Output: M_n, M_e
1 Cria o grafo aumentado $G^{s'}$
2 Resolve QVIA-SDN com variáveis relaxadas
3 **for** $k \in N^v$ **do**
4 **for** $z \in \Omega(k)$ **do**
5 $p_z = \alpha(\sum_{ij \in E^v} f_{ijkz} + f_{ijzk}) + (1-\alpha)x_{kz}$
6 **end**
7 $z_{max} = \text{argmax}\{p_z | z \in \Omega(k)\}$
8 **if** $z_{max} = \emptyset$ **then**
9 Rejeita G^v
10 **end**
11 $M_n(n) \leftarrow z_{max}$
12 **end**
13 **if** *BDC*$(G^v, G^{s'}, M_n, M_e)$ **then**
14 Atualizada capacidades residuáis dos recursos físicos
15 Return M_n, M_e
16 **else**
17 Rejeita G^v
18 **end**

Algorithm 1: QVIA-SDN baseado em [Chowdhury et al. 2012].

Input: $G^v, G^{s'}, M_n, M_e$
Output: Verdairo ou falso e os caminhos para M_e
1 **for** $ij \in E^v$ **do**
2 **if** $M_n(i) == M_n(j)$ **then**
3 **continue**
4 **end**
5 **for** $p \in P^s(i,j)$ **do**
6 $path \leftarrow \{\}$
7 $lat\_path \leftarrow 0$
8 **for** $u \in p$ **do**
9 **if** $e_{ijpu} > 0$ **then**
10 $lat\_path \leftarrow lat\_path + lat(u,c)$
11 $path \leftarrow path + u + controller(u)$
12 **else**
13 $path \leftarrow path + u$
14 **end**
15 **end**
16 **if** $R(path) \geq c_{ij} \wedge lat\_path \leq d_{ij}$ **then**
17 Set $M_e(ij) \leftarrow path$
18 **break**
19 **else**
20 $path = resolveMochila(path, d_{ij})$
21 **if** $path$ **then**
22 $M_e(ij) \leftarrow path$
23 **break**
24 **else**
25 Rejeita G^v
26 **end**
27 **end**
28 **end**
29 **end**

Algorithm 2: Busca Determinística de Caminhos.

No Alg. 1, baseado em [Chowdhury et al. 2012], é criado o grafo aumentado conectando os recursos virtuais a seus candidatos, de acordo com as definições da Seção 4.1. Então o LP é resolvido (linhas 1 e 2). Posteriormente, apenas um candidato adequado para alocar o recurso virtual é identificado. Para tal, p_z é calculado para os candidatos de k (linhas 3 a 11), dado pela soma ponderada de x_{kz} e o total de fluxo passando por kz em ambas as direções. Esta abordagem reconstrói a correlação entre f e x, identificando o mapeamento de vértices a arestas. O peso α indica a preferência (rede ou nós) durante esta etapa. O candidato com o maior p_z é selecionado para alocar o recurso virtual. Quando nenhum candidato é identificado, a alocação da IV é rejeitada (linhas 8 e 9).

Depois de identificar um mapeamento para MVs e *switches*, os caminhos físicos para alocar os enlaces virtuais são selecionados, considerando as particularidades do ambiente SDN (controladores, *switches* e tabelas de encaminhamento). A Busca Determinística de Caminhos (BDC) é guiada pela variável e, bem como por requisitos de QoS (latência e largura de banda) para identificar se um caminho físico é capaz de alocar um enlace virtual ij (Alg. 2). Além disso, busca reduzir a utilização dos *switches* através da

alocação de entradas de encaminhamento no controlador, quando possível. Quando dois recursos virtuais comunicantes são alocados no mesmo recurso físico, é assumido que o último possui capacidade suficiente para respeitar as restrições de largura de banda e latência (linhas 2 a 4). Todos os caminhos candidatos pré-selecionados para alocar um enlace virtual ij (linha 5) são analisados considerando a presença ou não de entradas na tabela de encaminhamento do controlador (linhas 8 a 14). Se existe algum valor em e, QVIA-SDN contabiliza a latência até o controlador ao invés de considerar a alocação na tabela de fluxo do *switch* correspondente.

Caminhos físicos com latência superior a requisitada são descartados. Ideia similar é aplicada em relação a largura de banda, ignorando aqueles caminhos que não atendem os requisitos mínimos. Entretanto, em alguns casos, o caminho pode ser reconfigurado para atender os requisitos de latência. Neste caso, resolvendo o problema da mochila, QVIA-SDN verifica se existe uma combinação de *switches* que pode alocar o enlace virtual respeitando os requisitos de QoS. Dentre os caminhos, são selecionados aqueles com o menor número de saltos e maior número de entradas de fluxo alocadas no controlador (reduzindo a carga do *switches*). Por fim, na ausência de caminhos físicos para alocar um enlace virtual, a IV é rejeitada.

6. Análise experimental

Os experimentos quantificam a utilização do *data center* na perspectiva do provedor, bem como a QoS percebida pelo cliente (latência) além da configuração da IV requisitada.

6.1. Métricas

Para representar os objetivos do provedor da nuvem (Seção 3.3), cinco métricas foram coletadas. (i) *Razão custo-revenue* fornece uma visão dos lucros do provedor ao alocar uma IV. (ii) *Fragmentação* do *data center* indica a porcentagem de recursos ativos, calculada através da divisão do número de recursos ativos pelo número total de recursos disponíveis. (iii) O *tempo médio de alocação* de uma requisição. (iv) A *taxa de aceitação* de requisições. (v) A *latência média* de uma IV, calculada pela soma das latências dos caminhos físicos que hospedam os enlaces virtuais, dividia pelo número de enlaces da IV.

6.2. Cenários de simulação

QVIA-SDN e um simulador de eventos discretos foram implementados em Java v1.8, utilizando CPLEX (v12.6.1.0) para resolver o LP. Os experimentos foram executados em um Intel Xeon E5-2620 2.0GHz - 24 núcleos, 256GB (DDR3) de RAM e 2TB. Para avaliar a aplicabilidade de QVIA-SDN em cenários de nuvem, a topologia *fat-tree* foi selecionada para o *data center*, enquanto para as IVs duas topologias comumente utilizadas são discutidas: multi-camadas e *Virtual Private Clouds (VPC)*.

Topologia *fat-tree* [Al-Fares et al. 2008]. Uma *fat-tree* é baseada em *switches* com k-portas. Assim, k *pods*, cada um contendo duas camadas de $k/2$ *switches*, são compostos, comportando até $k^3/4$ servidores. Neste trabalho, são consideradas duas configurações, $k = 4$ e $k = 8$. A capacidade dos servidores foi calculada atribuindo pesos a cada um dos recursos (CPU, RAM e armazenamento). Os pesos foram definidos empiricamente e representam a importância de cada recurso do processo de alocação de IVs. Assim, a CPU tem um peso de 50%, a RAM e o armazenamento 25% cada.

Tomando como base o servidor no qual a análise experimental foi executada, a capacidade ponderada dos servidores do *data center* é igual a 576. Quanto aos *switches* físicos (núcleo, agregação e aresta), a capacidade foi definida como 100 entradas nas tabelas de encaminhamento. A largura de banda entre os *switches* de núcleo e os *pods* é definida como 10Gbps, e 1Gbps para enlaces dentro dos *pods*. A latência entre qualquer par de recursos físicos é definida como 1ms, enquanto a latência entre os *switches* de núcleo e o controlador SDN é 2ms. Os *switches* de núcleo são diretamente conectados com o controlador SDN, enquanto os outros *switches* estão conectados por caminhos lógicos. A organização hierárquica das regiões e zonas na *fat-tree* é definida da seguinte maneira: cada *pod* representa uma zona, e cada par de *pods* representa uma região.

Requisições de IVs multi-camadas (NC). Diversos inquilinos organizam suas IVs em multi-camadas [Jennings and Stadler 2014], incluindo um balanceador de carga, encarregado de distribuir as requisições para um conjunto de servidores, que eventualmente consultam um banco de dados. Para simulação, cada camada possui um *switch* virtual que solicita QoS de rede, além das MVs. A distribuição de MVs ocorre da seguinte forma: um balanceador de carga, quatro servidores e quatro bancos de dados.

Requisições de IVs VPC. Amazon EC2 introduziu o provisionamento de VPCs[1], que é composta por um conjunto de regras de acesso e um conjunto de MVs, compondo uma rede privada, gerenciada pelo cliente. Para compor as requisições VPC, um conjunto de 9 MVs é conectado a um *switch* SDN, representando as regras de acesso.

As capacidades dos recursos virtuais segue as instâncias M3 da Amazon EC2[2], nas configurações *medium*, *large*, *xlarge* e *2xlarge*. A capacidade das MVs é calculada utilizando os mesmo pesos aplicados para o *data center*, obtendo os valores 2, 10, 25 e 51 para cada configuração, respectivamente. Já a capacidade dos enlaces virtuais segue uma distribuição normalizada entre 10, 20, 40 e 80% das larguras de bandas média observadas por cada instância [Persico et al. 2015]: 492, 739, 958 e 1188 Mbps, respectivamente. Quanto a localização geográfica, todas as IVs recebem uma região, porém existe só 50% de chances de uma IV especificar uma zona. Um conjunto com 50 requisições (VPC ou multicamadas) igualmente distribuído entre os quatro tipo de instâncias M3 é submetido para cada cenário físico. O tempo de chegada de cada IV é uniformemente distribuído entre 100 intervalos discretos, com tempo ativo máximo de 30 intervalos. A latência atual, bem como a capacidade residual dos recursos são coletadas a cada requisição submetida.

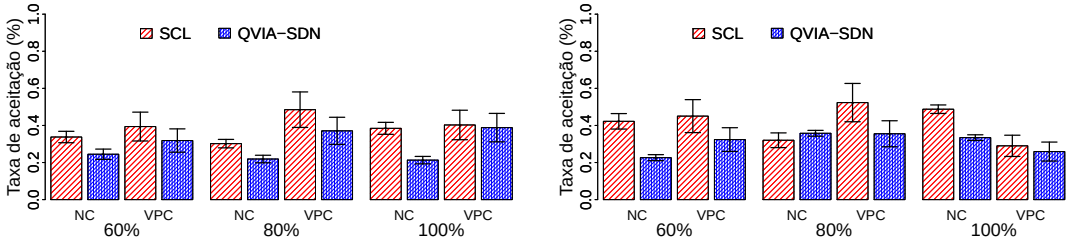
6.3. Resultados da simulação

Os resultados mostram médias com 95% de intervalo de confiança. QVIA-SDN é comparado com um algoritmo base sem controle de latência (SCL) baseado na proposta de [Chowdhury et al. 2012]. Baseado em observações empíricas, $\alpha = 0.9$ (Seção 5.3) e $\beta_u = \gamma_e = \alpha_{uv} = 1$ (Seção 4.2). Dois cenários são definidos variando o tipo de IV.

Taxa de aceitação. Os resultados para $k = 4$ e $k = 8$ são apresentados na Figura 3. A Figura 3(a) indica que diante de poucos recursos é possível alocar um número maior de VPCs. Para o cenário com $k = 8$ houve um equivalência entre os dois tipos de IVs. Sobretudo, QVIA-SDN e SCL possuem taxas de aceitação equivalentes, porém QVIA-SDN fornece uma melhora de QoS na perspectiva do inquilino.

[1]Virtual Private Cloud (VPC): `https://aws.amazon.com/vpc/`.

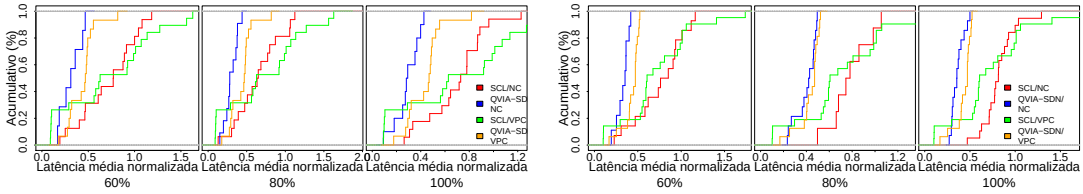
[2]Instâncias M3 - Amazon EC2: `https://aws.amazon.com/pt/ec2/instance-types/`



(a) $k = 4$. (b) $k = 8$.

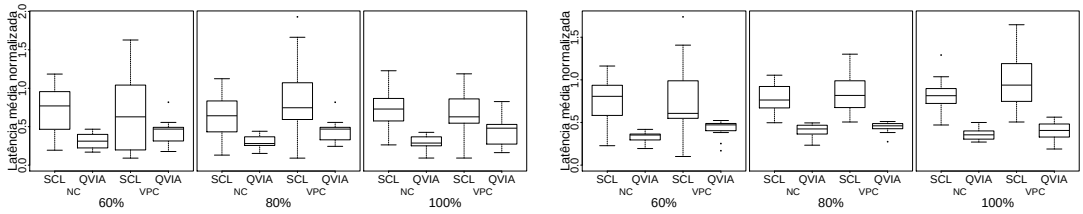
Figura 3. Taxa de aceitação.

Latência média. A Figura 4 mostra a distribuição acumulada da latência média normalizada. A latência experimentada nas IVs de NC foi inferior aquela presente nas IVs de VPC, ou seja, requisições NCs ocuparam mais entradas nos *switches*, justificando uma menor taxa de aceitação. Em todos os cenários QVIA-SDN ofereceu menor latência e variabilidade quando comparado ao algoritmo SCL (Figura 5).



(a) $k = 4$. (b) $k = 8$.

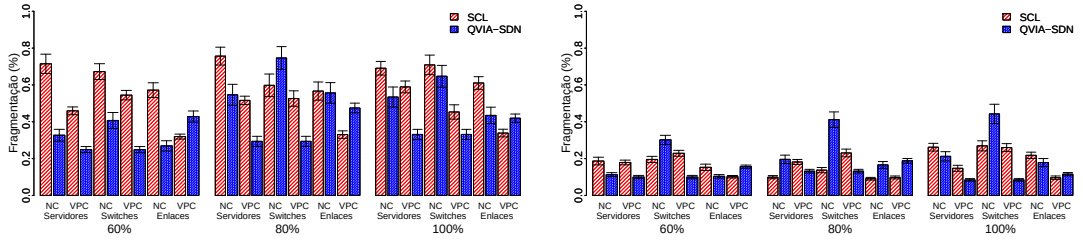
Figura 4. Distribuição acumulada da latência média normalizada.



(a) $k = 4$. (b) $k = 8$.

Figura 5. Variabilidade da latência média.

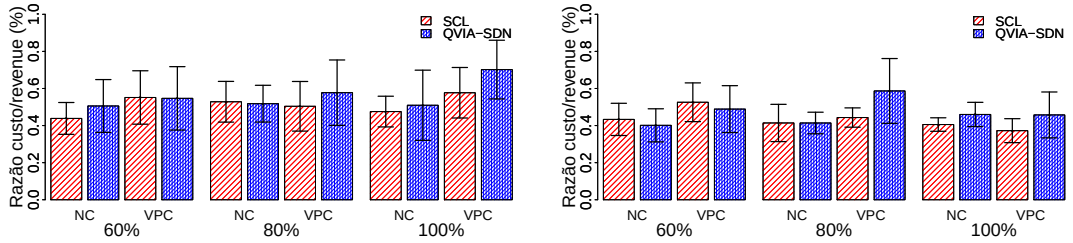
Fragmentação. As Figuras 6(a) e 6(b) mostram a fragmentação para $k = 4$ e $k = 8$, respectivamente. Como esperado, requisições NC utilizam mais recursos de comunicação e requisições VPC permitem uma maior consolidação de servidores, o que também impactou nas taxas de aceitação. Ainda, QVIA-SDN além de melhorar a latência experimentada pelo inquilino reduziu a fragmentação do *data center* (exceto *switches* em NC).



(a) $k = 4$. (b) $k = 8$.

Figura 6. Fragmentação do *data center*.

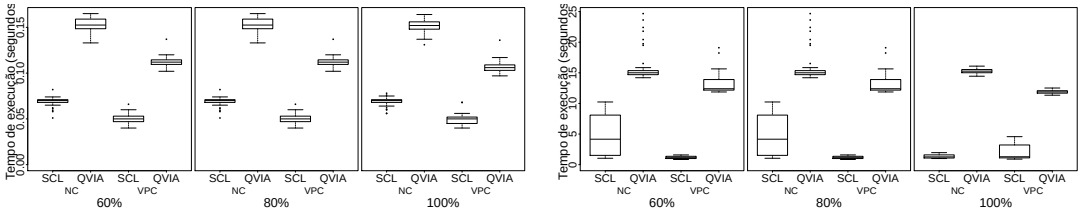
Relação entre custo e *revenue*. As Figuras 7(a) e 7(b) indicam que existe uma equivalência entre QVIA-SDN e SCL e entre NC e VPC. Ainda, é importante realizar uma relação com a taxa de aceitação. SCL obteve maior aceitação, porém uma menor razão custo-*revenue*. Ou seja, os cenários com QVIA-SDN conseguiram alocar IVs com cargas maiores do que aquelas alocadas por SCL. O mesmo raciocínio pode ser aplicado a VPC em relação a NC. Por fim, usando QVIA-SDN é possível melhorar a latência na perspectiva do inquilino sem prejudicar as métricas que representam a perspectiva do provedor.



(a) $k = 4$. (b) $k = 8$.

Figura 7. Relação entre custo e *revenue*.

Tempo médio de alocação de IVs. As Figuras 8(a) e 8(b) mostram que o tempo médio para alocação de IVs com QVIA-SDN é superior na maioria dos casos, entretanto, inferior a 30 segundos no pior caso (para os cenários com $k = 8$).



(a) $k = 4$. (b) $k = 8$.

Figura 8. Tempo médio de alocação de IVs.

7. Conclusões

SDN introduziu benefícios e desafios no provisionamento de IVs em ambientes de nuvens IaaS. O presente trabalho fornece uma formulação baseada em MIP para alocação de IVs em ambiente SDN considerando as restrições impostas. Em seguida, variáveis do MIP foram relaxadas obtendo um LP. O mecanismo proposto, QVIA-SDN, combina o LP com heurísticas de arredondamento. QVIA-SDN, foi comparado com um mecanismo de base, considerando dois tipos de requisições de IVs (NC e VPC). Os resultados obtidos indicam que em *data centers* baseados em SDN é possível realizar a alocação de IVs com garantias de QoS sem afetar as métricas do provedor. Ainda, as técnicas de redução de candidatos, indicam que nem todos os candidatos precisam ser considerados no processo de alocação de IVs. Os resultados promissores, abrem caminhos para trabalhos futuros. Uma primeira linha de pesquisa pode explorar o conhecimento centralizado do controlador para realizar o compartilhamento de recursos de largura de banda residual, enquanto outra linha indica a implementação em um *framework* de gerenciamento.

Agradecimentos: ao laboratório LabP2D, programa PROMOP/UDESC e FAPESC.

Referências

Al-Fares, M., Loukissas, A., and Vahdat, A. (2008). A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74.

Al-Shabibi, A., De Leenheer, M., Gerola, M., Koshibe, A., Parulkar, G., Salvadori, E., and Snow, B. (2014). Openvirtex: Make your virtual sdns programmable. In *Proc. of the HotSDN '14*, pages 25–30. ACM.

Ballani, H., Costa, P., Karagiannis, T., and Rowstron, A. (2011). Towards predictable datacenter networks. *SIGCOMM Comput. Commun. Rev.*, 41(4):242–253.

Cavalcanti, G., Obelheiro, R., and Koslovski, G. (2014). Optimal resource allocation for survivable virtual infrastructures. In *IEEE Conference on Design of Reliable Communication Networks*, pages 1–8.

Chowdhury, M., Rahman, M., and Boutaba, R. (2012). Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *Networking, IEEE/ACM Transactions on*, 20(1):206–219.

de Oliveira, R. and Koslovski, G. P. (2017). A tree-based algorithm for virtual infrastructure allocation with joint virtual machine and network requirements. *International Journal of Network Management*, 27(1):e1958–n/a. e1958 nem.1958.

Demirci, M. and Ammar, M. (2014). Design and analysis of techniques for mapping virtual networks to software-defined network substrates. *Computer Communications*, 45:1 – 10.

Fischer, A., Botero, J., Till Beck, M., de Meer, H., and Hesselbach, X. (2013). Virtual network embedding: A survey. *Communications Surveys Tutorials, IEEE*, 15(4):1888–1906.

Jennings, B. and Stadler, R. (2014). Resource management in clouds: Survey and research challenges. *Journal of Network and Systems Management*, pages 1–53.

Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C., Azodolmolky, S., and Uhlig, S. (2015). Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1):14–76.

Manvi, S. S. and Shyam, G. K. (2014). Resource management for infrastructure as a service (iaas) in cloud computing: A survey. *Journal of Network and Computer Applications*, 41:424 – 440.

Mell, P. M. and Grance, T. (2011). SP 800-145. The NIST Definition of Cloud Computing. Technical report, NIST, Gaithersburg, MD, United States.

Mijumbi, R., Serrat, J., Gorricho, J. L., and Boutaba, R. (2015). A path generation approach to embedding of virtual networks. *IEEE Transactions on Network and Service Management*, 12(3):334–348.

Mijumbi, R., Serrat, J., Rubio-Loyola, J., Bouten, N., De Turck, F., and Latre, S. (2014). Dynamic resource management in sdn-based virtualized networks. In *Int. Conf. on Network and Service Management*.

Persico, V., Marchetta, P., Botta, A., and Pescape, A. (2015). Measuring network throughput in the cloud: The case of amazon ec2. *Computer Networks*, 93:408 – 422. Cloud Networking and Comm. {II}.

Popa, L., Krishnamurthy, A., Ratnasamy, S., and Stoica, I. (2011). Faircloud: Sharing the network in cloud computing. In *Proc. of the Workshop on Hot Topics in Networks*, pages 22:1–22:6. ACM.

Rost, M., Fuerst, C., and Schmid, S. (2015). Beyond the stars: Revisiting virtual cluster embeddings. In *In Proc. ACM SIGCOMM Computer Communication Review*.

Sherwood, R., Gibb, G., kiong Yap, K., Casado, M., Mckeown, N., and Parulkar, G. (2009). Flowvisor: A network virtualization layer. Technical report, Deutsche Telekom, Stanford University, Nicira Networks.

Stallings, W. (2015). *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud*. Addison-Wesley Professional, 1st edition.

Tao, F., Jun, B., and Ke, W. (2015). Allocation and scheduling of network resource for multiple control applications in sdn. *Communications, China*, 12(6):85–95.

Caracterizando Estratégias de Domínio Espacial para Gerenciamento de Regras em Redes Definidas por Software

Gustavo de Araújo[1], Marcelo Marotta[1], Juliano Wickboldt[1], Cristiano Both[2], Luciano Gaspary[1], Juergen Rochol[1], Lisandro Granville[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul

[2]Departamento de Matemática Aplicada e Ciências Sociais
Universidade Federal das Ciências da Saúde de Porto Alegre

{gustavo.araujo, mamarotta, jwickboldt}@inf.ufrgs.br

cbboth@ufcspa.edu.br, {paschoal, juergen, granville}@inf.ufrgs.br

Abstract. *Software-Defined Networks (SDN) based on Openflow protocol perform data forwarding through flow tables using match/action mechanisms. These tables have limited rules storage capacity, restricting network scalability and performance. Considering these restrictions, the main strategies for managing spatial domain rules, i.e., flow agregation and multiple flow tables, are promising to use the available storage space of the tables. These strategies impact packet processing in different ways, influencing the performance of the forwarding devices and the network. In spite of the impact, the comparison between flow aggregation and multiple flow tables is poorly exploited, leaving open the definition of which strategy is more appropriate for a network considering its topology and workload. In this article a quantitative characterization is proposed, defining the gains brought by these spatial domain strategies for forwarding devices and in different topologies.*

Resumo. *As Redes Definidas por Software (SDN) baseadas no protocolo Openflow realizam o encaminhamento de dados por meio de tabelas de fluxos utilizando mecanismos de match/action. Tais tabelas possuem uma capacidade de armazenamento de regras limitada, restringindo a escalabilidade e o desempenho da rede. Considerando essas restrições, as principais estratégias de gerenciamento de regras de domínio espacial, i.e., agregação de regras e múltiplas tabelas, mostram-se promissoras na utilização do espaço de armazenamento disponível das tabelas. Essas estratégias impactam de diferentes maneiras no processamento dos pacotes influenciando o desempenho dos dispositivos de encaminhamento e, de forma mais geral, da rede. Apesar do impacto gerado, a comparação entre as estratégias de agregação de fluxos e múltiplas tabelas é fracamente explorada, deixando em aberto a definição de qual estratégia é mais adequada para uma rede, considerando sua topologia e carga de trabalho. Neste artigo realiza-se uma caracterização quantitativa definindo os ganhos trazidos por essas estratégias de domínio espacial para dispositivos de encaminhamento e em diferentes topologias.*

1. Introdução

Os conceitos inerentes às Redes Definidas por Software (SDN) mostram-se fundamentais na evolução das redes de computadores e continuam sendo investigados tanto pela industria quanto pela academia [Wickboldt et al. 2015]. Entre os principais conceitos destacam-se: o encaminhamento de tráfego baseado em fluxos e a abstração da lógica de controle para uma entidade em software, chamada controlador. Através desses conceitos, um dispositivo de encaminhamento torna-se capaz de desempenhar diferentes funções, por exemplo, controlar o acesso a um servidor de aplicação ou balancear carga entre diferentes enlaces [Kreutz et al. 2015]. As funções de rede desempenhadas pelos dispositivos são definidas a partir de regras de encaminhamento que são armazenadas em tabelas de fluxos. Tais tabelas possuem uma capacidade de armazenamento de regras limitada, a qual é várias ordens de grandeza menor que o necessário para a operação de determinados tipos de rede como, por exemplo, *backbones* e redes de *datacenters* [Nguyen et al. 2016]. Portanto, o adequado gerenciamento de regras se torna um requisito para a escalabilidade das redes SDN.

Levando em consideração a escalabilidade das redes SDN, estratégias para gerenciamento de regras de encaminhamento foram propostas na literatura. Tais estratégias podem ser classificadas em dois grupos: (*i*) domínio temporal e (*ii*) domínio espacial. As estratégias de domínio temporal objetivam a deleção de regras da tabela de fluxos a partir de uma determinada política (*e.g., Least Recently Used* e *timeouts* dinâmicos e estáticos) [Neves et al. 2016]. Para compor o estado dos fluxos e realizar a deleção, o controlador precisa consultar os dispositivos de encaminhamento de tempos em tempos (*polling*), a fim de obter em tempo real, as informações de utilização de cada fluxo (*e.g.,* duração e número de pacotes recebidos). Esse *polling* insere uma sobrecarga significativa no tráfego do canal de controle, no processamento do controlador e nos dispositivos de encaminhamento. A sobrecarga gerada torna as estratégias de domínio temporal não escaláveis, mesmo para pequenas redes de *datacenters* [Vishnoi et al. 2014].

As estratégias de domínio espacial, por sua vez, objetivam a representação da maior quantidade de fluxos com o menor número de regras de encaminhamento na rede. Esta representação é processada pelo controlador considerando diferentes abordagens de gerenciamento, das quais destacam-se, (*a*) agregação de fluxos e (*b*) múltiplas tabelas. O gerenciamento por agregação de fluxos permite a representação de dois ou mais fluxo utilizando uma única regra, sem alterar a semântica de encaminhamento [Kamiyama et al. 2014]. Já, o gerenciamento por múltiplas tabelas permite que um grupo de fluxos tenha sua semântica dividida, sendo representada em poucas regras pertencentes a duas ou mais tabelas de acordo com as políticas empregadas (*e.g.,* uma tabela para encaminhamento de portas e outra para controle de acesso) [ONF 2015]. As estrategias de domínio espacial, mostram-se promissoras na utilização do espaço de armazenamento disponível nas tabelas de fluxos dos dispositivos de encaminhamento, pois necessitam de um monitoramento menor ou, em alguns casos, nenhum, quando comparadas as estratégias de domínio temporal.

O gerenciamento por agregação e múltiplas tabelas possuem seus próprios *modus operandi*, impactando no desempenho de cada dispositivo de encaminhamento e, de forma geral, da rede. Esse impacto pode ser mensurado a partir das métricas de desempenho, como o número de regras de encaminhamento, a quantidade de intervenções do controla-

dor, a latência e o *jitter* da rede. Entretanto, existem questionamentos sobre o impacto no emprego de cada uma dessas formas de gerenciamento, o que agrava-se para redes com diferentes topologias, cargas de trabalho e políticas empregadas. Por exemplo, gerenciamento por agregação de fluxos precisa adicionar e remover regras com maior frequência comparada ao gerenciamento por múltiplas tabelas, aumentando latência e *jitter*. Já, o gerenciamento por múltiplas tabelas precisa percorrer um número maior de tabelas na definição da ação a ser tomada comparada ao gerenciamento por agregação, também incorrendo latência ou *jitter*. Como pode ser visto, ambas as formas de gerenciamento apresentam impactos diferentes para as mesmas métricas. Logo, uma comparação entre gerenciamento por agregação e por múltiplas tabelas torna-se fundamental na definição de qual delas é a mais adequada para ser empregada em uma rede. No melhor conhecimento, não existe um trabalho comparativo que investigue ambas as formas de gerenciamento espacial, permanecendo uma questão de pesquisa em aberto.

Neste artigo é proposto uma comparação quantitativa entre os gerenciamentos por agregação e múltiplas tabelas em três diferentes topologias de rede. Primeiramente, busca-se analisar cada dispositivo individualmente, verificando a redução na quantidade de regras que cada forma de gerenciamento de domínio espacial obtêm. Em seguida, é analisada a quantidade de intervenções que o controlador realiza para cada forma de gerenciamento. Para uma comparação completa das formas de gerenciamento, utiliza-se como linha base a operação padrão para encaminhamento de camada 2 em SDN baseada em OpenFlow. Além disso, para avaliar a influência que o gerenciamento traz para diferentes redes, utilizou-se um ambiente de redes de topologia variadas: único salto, anel e árvore. Os resultados mostram que um gerenciamento por agregação de fluxos pode reduzir drasticamente (aproximadamente 95%) a quantidade regras de encaminhamento independente da topologia de rede. Entretanto, a quantidade de intervenções com o controlador mantém-se extremamente alta (acima de 370%), bem como o *jitter*, onde o mesmo apresenta um valor duas vezes superior a linha base, para uma topologia de anel. O gerenciamento por múltiplas tabelas possui o melhor custo-benefício, levando-se em consideração a quantidade de regras instaladas e a quantidade de intervenções necessárias do controlador, apresentando baixa latência e *jitter*, em qualquer topologia de rede.

O restante do artigo está organizado da seguinte maneira. A Seção 2 discute os trabalhos relacionados. A Seção 3 explora e exemplifica as estratégias de gerência de domínio espacial. A Seção 4, apresenta os resultados obtidos. Finalmente, a Seção 5 expõe as principais conclusões do estudo e perspectivas de trabalhos futuros.

2. *Background* e Trabalhos Relacionados

A limitada capacidade de armazenamento dos dispositivos de encaminhamento das SDNs motivou o avanço na área de gerenciamento de regras das tabelas de fluxos. Nessa área, destacam-se as diferentes formas de gerenciamento sendo classificadas em dois grupos: aquelas pertencentes ao domínio temporal e espacial. Desta forma, na Subseção 2.1, descreve-se o gerenciamento de domínio temporal e suas limitações. Já, na Subseção 2.2, as diferentes formas de se utilizar o gerenciamento de domínio espacial são apresentadas.

2.1. Gerenciamento de Domínio Temporal

O gerenciamento de domínio temporal consiste em remover uma regra da tabela de fluxos, após um determinado período de tempo. Essa remoção de regras pode ser realizada

por algoritmos como *Least Recently Used* (LRU), *First In First Out* (FIFO) ou remoção randômica. A escolha do algoritmo de remoção é determinante para o desempenho da rede [Lee et al. 2013]. Por exemplo, a utilização de um algoritmo FIFO removerá as regras que foram instaladas a mais tempo na tabela de fluxos. Esta remoção pode ser realizada de maneira pro-ativa, com o próprio dispositivo de encaminhamento removendo as regras, depois de um período determinado de tempo (*hard timeout*), ou de um período de inatividade do fluxo (*idle timeout*). Entretanto, os fluxos possuem durações variadas que podem tanto ser instantâneas ou maiores que o *hard timeout*, levando a remoção tardia ou indevida das regras. Essas remoções implicam na utilização ineficiente da capacidade disponível de armazenamento [Benson et al. 2010].

Um estudo caracteriza e compara diferentes propostas de gerenciamento de regras de domínio temporal [Neves et al. 2016]. Nessa comparação, as formas de gerenciamento temporal, *i.e., idle timeouts* incrementais adaptativos, remoção probabilística de regras, *hard timeouts* adaptativos, apresentaram, no melhor dos casos, uma utilização da capacidade de armazenamento de regras 15% maior que o caso ótimo e degradando-se (aumentando) gradualmente para os demais casos. Adicionalmente, foi constatado que pequenas mudanças nas características do tráfego (*e.g.,* tempo de duração dos fluxos) afetam consideravelmente o desempenho do gerenciamento de domínio temporal. Em síntese, o gerenciamento de domínio temporal mostra-se não escalável às diferentes redes e resiliente às mudanças nas características das mesmas. Nas redes onde o gerenciamento de domínio temporal mostra-se inadequado, o gerenciamento de domínio espacial pode ser uma potencial solução, sendo o tema deste trabalho e da próxima subseção.

2.2. Gerenciamento de Domínio Espacial

As formas de gerenciamento de domínio espacial podem ser classificadas em dois grupos: (*i*) *inter-switch* e (*ii*) *intra-switch*. No primeiro grupo, o gerenciamento é considerado *inter-switch*, pois as regras são gerenciadas considerando o encaminhamento e a semântica dos fluxos dentro de um conjunto de dispositivos da rede. No segundo grupo, o gerenciamento é considerado *intra-switch*, pois as regras são gerenciadas considerando apenas um dispositivo de encaminhamento e os fluxos que passam por ele.

O gerenciamento *inter-switch*, também denominado como posicionamento de regras, consiste em dividir um conjunto de regras de encaminhamento e distribuí-los entre os dispositivos da rede. Esse tipo de gerenciamento espacial é normalmente modelado como um problema de otimização que deve decidir quais regras devem ser instaladas em cada dispositivo. A função objetivo de otimização depende da aplicação que pretende-se implementar, por exemplo, minimizar a quantidade total de regras instaladas na rede [Kanizo et al. 2013] ou minimizar a energia consumida [Giroire et al. 2014]. Esse gerenciamento pode ocasionar duplicação de regras e sobrecarga no processamento para execução do algoritmo de otimização [Nguyen et al. 2016]. Além disso, este gerenciamento precisa de um monitoramento de fluxos frequente, levando aos mesmos contrapontos do domínio temporal. Assim, o gerenciamento *inter-switch* não é foco deste trabalho.

As principais formas de gerenciamento de domínio espacial *intra-switch* são: (*i*) agregação de fluxos e (*ii*) múltiplas tabelas. Na primeira, o gerenciamento por agregação de fluxos opera a partir da identificação das regras que possuem a mesma ação (*e.g.,* encaminhar pacote para uma determinada porta). Em seguida, identifica-se os campos de correspondência (*match*) das regras que possuem similaridades (*e.g.,* IPs de origem que

pertencem a mesma subrede), para serem agregadas sob uma nova regra única (*e.g.,* todos os IPs origem de uma subrede serão encaminhados para uma mesma porta). Desta forma, uma regra pode representar diversos fluxos, reduzindo, assim, a quantidade de regras da tabela. Agregação de fluxos é uma estratégia tradicional para reduzir a quantidade de regras em tabelas de roteamento das atuais redes IP e motivado pelo bom desempenho obtido, veem sendo aplicada em redes SDN/Openflow [Nguyen et al. 2016].

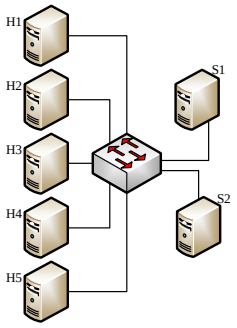
Na segunda forma, o gerenciamento por múltiplas tabelas opera a partir da identificação da semântica de um fluxo (*e.g.,* fluxo com destino a uma interface virtual a ser encaminhado por uma determinada porta), processando sua divisão para duas ou mais regras. A partir dessa divisão, uma nova regra é criada com uma semântica simplificada e armazenada em uma tabela que a represente. Dessa forma, novos fluxos com semânticas compostas são representados por regras já instaladas com semânticas mais simples, utilizando um número menor de regras armazenadas no dispositivo. Gerenciamento por múltiplas tabelas é previsto desde a versão 1.1 do protocolo OpenFlow, sendo a atual forma de gerenciamento incentivada pela *Open Network Foundation* (ONF) [ONF 2015]. Sua utilização adiciona mais complexidade ao processo de busca na tabela de fluxos (*lookup*) para dispositivos OpenFlow que implementam tabelas de fluxos em hardware. Gerenciamento por múltiplas tabelas é pouco explorada pela literatura, embora apresente grande potencial para reduzir a quantidade de regras utilizadas.

As diferentes formas de gerenciamento espacial *intra-switch*, mostram-se promissoras, por não possuírem a necessidade de um frequente monitoramento do estado atual das tabelas de fluxos. Entretanto, a comparação entre essas formas de gerenciamento é fracamente explorada na literatura, impossibilitando a definição de qual forma de gerenciamento *intra-switch* é a mais adequada para uma determinada rede. Na próxima seção, explora-se como cada uma das formas de gerenciamento de domínio espacial *intra-switch* podem ser aplicadas, para posteriormente serem comparadas em uma rede OpenFlow.

3. Explorando as estratégias de domínio espacial

As formas de gerenciamento de domínio espacial *intra-switch* podem ser comparadas quando aplicadas a uma rede baseada em OpenFlow, onde políticas definem a semântica do encaminhamento de fluxos para a correta operação de uma aplicação de rede. Dessa forma, utiliza-se como exemplo, a implementação de políticas de controle de acesso a servidores de aplicação.

Regras de controle de acesso definem quais fluxos são autorizados a acessar determinados serviços ou nodos da rede. Essas regras são implementadas associando endereços IP aos serviços disponíveis. Idealmente, todas as regras que implementam controle de acesso devem estar presentes no último salto antes do serviço que pretende-se acessar. Desta maneira, evita-se o processamento de regras desnecessárias em dispositivos que possuem a função exclusiva de encaminhamento de pacotes. Entretanto, com a limitação de memória existente em dispositivos de encaminhamento baseado em OpenFlow, posicionar todas as regras em um único dispositivo é impraticável para redes com muitos usuários [Nguyen et al. 2016]. Além disso, conforme a quantidade de regras na tabela de fluxos aumenta, mais processamento é exigido por parte do dispositivo de encaminhamento para realizar o *lookup*. Esse processamento afeta negativamente o desempenho da rede fazendo com que a latência e o *jitter* aumentem. Portanto, o gerenciamento de regras, principalmente, de domínio espacial *intra-switch* torna-se uma exigência.



(a) LAN com cinco clientes e dois servidores de aplicação conectados por um *switch*

Tabela única		
#	Regra	Ação
R1	Se ip.src=H1 e ip.dst=S1	Encaminha na porta P6
R2	Se ip.src=H1 e ip.dst=S2	Descartar pacote
R3	Se ip.src=H2 e ip.dst=S1	Encaminhar na porta P6
R4	Se ip.src=H2 e ip.dst=S2	Descartar pacote
R5	Se ip.src=H3 e ip.dst=S1	Descartar pacote
R6	Se ip.src=H1 e IP.dst=H2	Encaminha na porta P2
R7	Se ip.src=H3 e ip.dst=H2	Encaminhar na porta P2
R8	Se Ip.src=S1 e ip.dst=H2	Encaminha na porta P2
R...	...	...
Rn	Regra padrão	Encaminhar para Controlador

(b) Representação das regras de encaminhamento com uma única tabela

Tabela agregando fluxos		
#	Regras	Ação
R1	Se ip.src=H1 e ip.dst=S1	Encaminha na porta P6
R2	Se ip.src=H1 e ip.dst=S2	Descartar pacote
R3	Se ip.src=H2 e ip.dst=S1	Encaminhar na porta P6
R4	Se ip.src=H2 e ip.dst=S2	Descartar pacote
R678	Se IP.dst=H2	Encaminhar pela porta 2
R...	...	...
Rn	Regra padrão	Encaminhar para Controlador

(c) Representação de uma regra de encaminhamento agregada

Tabela 1: Controle de acesso	
Regra	Ação
Se ip.src=H1 e ip.dst=S2	Descartar pacotes
Se ip.src=H2 e ip.dst=S2	Descartar pacotes
Se ip.src=H3 e ip.dst=S1	Descartar pacotes
Se ip.src=H4 e ip.dst=S1	Descartar pacotes
Se ip.src=H5 e ip.dst=S1	Descartar pacotes
Regra padrão	Encaminhar para tabela 2

Tabela 2: Encaminhamento	
Regras	Ação
Se ip.dst=S1	Encaminha na porta P6
Se ip.dst=S2	Encaminha na porta P7
Se ip.dst=H1	Encaminha na porta P1
Se ip.dst=H2	Encaminha na porta 2
Se ip.dst=H3	Encaminha na porta P3
Se ip.dst=H4	Encaminha na porta P4
Se ip.dst=H5	Encaminha na porta P5
Regra padrão	Enviar para controlador

(d) Representação das regras de encaminhamento com múltiplas tabelas de fluxos

Figura 1. Exemplo de regras de encaminhamento

Para exemplificação, apresenta-se o cenário na Figura 1(a), uma pequena LAN com um único dispositivo de encaminhamento conectando cinco clientes (H1, H2, H3, H4 e H5) a dois servidores de aplicação (S1 e S2). O dispositivo de encaminhamento possui as funções de: (*i*) realizar o controle de acesso aos servidores e (*ii*) encaminhar os pacotes entre os nodos da rede. Em termos de controle de acesso, o dispositivo de encaminhamento autoriza ou rejeita os fluxos dentro da rede de acordo com as seguintes políticas: (*a*) os clientes H1 e H2 têm acesso apenas ao servidor S1, (*b*) os clientes H3 e H4 têm acesso apenas ao servidor S2, (*c*) o cliente H5 possui acesso a ambos os servidores e, finalmente, (*d*) os clientes possuem acesso um ao outro. Em termos de encaminhamento, cada um dos fluxos identificados pode ser redirecionado para uma das portas, onde um determinado nodo final se encontra (*i.e.,* H1:P1, H2:P2, H3:P3 ... S1:P6, S2:P7).

A Figura 1(b) apresenta uma tabela de fluxos com regras de encaminhamento compostas por dois campos de cabeçalho para realizar o processo de *lookup*: IP origem e IP destino. As ações que podem ser associadas a cada regra são: descartar pacotes caso um fluxo não pertença a lista de clientes autorizados ou encaminhar os pacotes dos fluxos para o destino, caso seja permitido. Para implementar o controle de acesso em apenas uma única tabela de fluxos, é necessária uma quantidade de regras que representem todas as combinações possíveis para a comunicação entre clientes e clientes e servidores de aplicação. Portanto, a quantidade de regras necessárias é de $O(n^2)$, sendo n a quantidade total de nodos na rede. Essa quantidade de regras pode ser maior que a capacidade de boa parte dos dispositivos de encaminhamento disponíveis no mercado [Costa et al. 2016].

A Figura 1(c) ilustra um exemplo de como pode ser realizada o gerenciamento por

agregação de fluxos. Nessa forma de gerenciamento, pretende-se mesclar duas ou mais regras de encaminhamento, a fim de representar vários fluxos em uma única regra. Por exemplo: H1, H3 e S1 desejam comunicar-se com H2. Nesse gerenciamento, as regras R6, R7 e R8 que possuem o mesmo destino e a mesma ação associada são mescladas em uma única regra. Assim, com uma única regra é possível representar três fluxos agregados. O gerenciamento por agregação pode reduzir drasticamente a quantidade de regras necessárias para representar fluxos em uma tabela OpenFlow. Por outro lado, esse gerenciamento mescla os fluxos impossibilitando que os mesmos sejam monitorados individualmente de uma forma precisa [Nguyen et al. 2015]. Outro aspecto a ser considerado é a carga de trabalho adicional que um algoritmo de agregação insere no processamento do controlador. Para agregar as regras de encaminhamento, o controlador deve ler o estado atual da tabela de fluxos do dispositivo, realizar a agregação e substituir o conjunto de regras originais pelo conjunto de regras agregadas. Esse processamento pode ocasionar perdas de pacotes, *loops* de encaminhamento e atrasos na rede [Luo et al. 2014].

Na Figura 1(d) pode-se observar o gerenciamento das regras por múltiplas tabelas de fluxos. Nessa forma de gerenciamento, subdivide-se uma tabela de fluxos em duas ou mais tabelas. Cada uma dessas sub-tabelas agrupa um conjunto de regras pertencentes a uma semântica, normalmente, determinada por uma política. A distribuição de regras e sequência pela qual os pacotes são analisados depende da aplicação que se pretende implementar. Seguindo o exemplo de controle de acesso do cenário apresentado, a primeira tabela armazena as regras referentes ao controle de acesso aos servidores. Essas regras são implementadas na forma de uma lista negra, *i.e.,* caso o fluxo pertença a lista, seus pacotes são descartados. Caso contrário, a tabela possui uma regra padrão que é aplicada aos pacotes que não encontrarem nenhuma entrada na lista correspondente. Nesse exemplo, a regra padrão é encaminhar os pacotes para a próxima tabela que armazena as regras de encaminhamento. Nessa segunda tabela, os pacotes pertencentes a cada um dos fluxos são encaminhados para portas de destino do dispositivo de encaminhamento.

As formas de gerenciamento de domínio espacial *intra-switch* propõem diferentes maneiras de se melhorar a utilização da capacidade de armazenamento limitado de entradas nas tabelas de fluxos, através da redução do conjunto de regras utilizadas pelos dispositivos de encaminhamento. Uma comparação entre estas formas de gerenciamento permite a identificação de qual forma de gerenciamento espacial é a mais indicada para uma rede com diferentes características, por exemplo, sua topologia, sendo o enfoque da próxima seção.

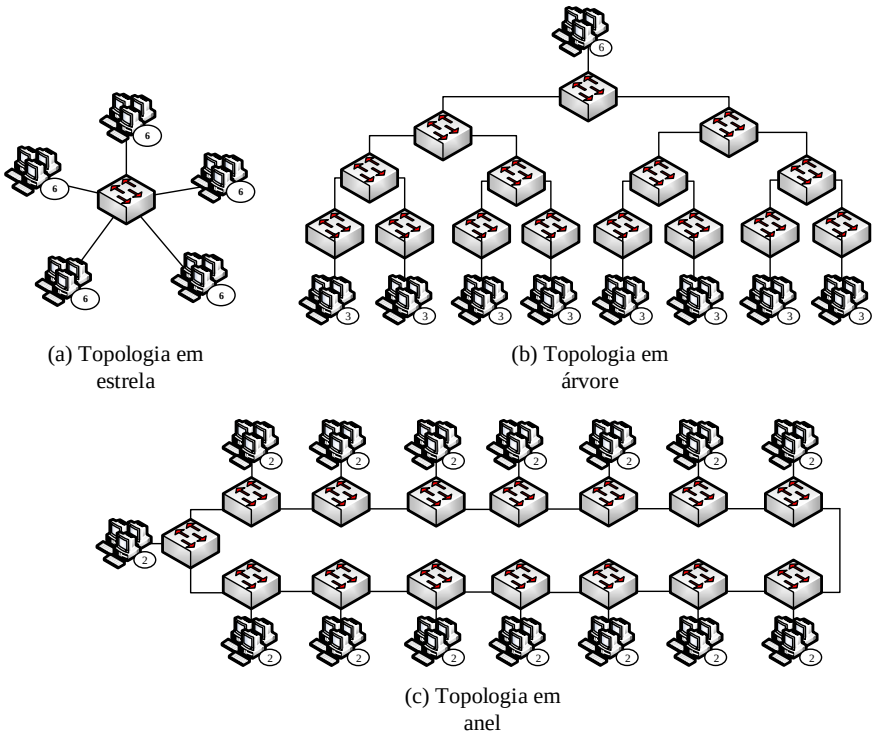
4. Comparação entre formas de gerenciamento de domínio espacial

Nessa seção apresenta-se a metodologia necessária para a realização da comparação entre as diferentes formas de gerenciamento (Subseção 4.1). Baseado nessa metodologia, na Subseção 4.2, discute-se os resultados obtidos.

4.1. Metodologia

Cenário. Para se comparar as diferentes formas de gerenciamento, três topologias são propostas: (*i*) estrela, (*ii*) árvore e (*iii*) anel. Na topologia de estrela, um único dispositivo de encaminhamento é utilizado para comunicar 30 nodos finais com o intuito de avaliar o impacto das formas de gerenciamento nesse dispositivo individualmente, como apresentado na Figura 2(a). Já, para realizar uma análise do impacto da utilização das diferentes

formas de gerenciamento de domínio espacial em um escopo mais amplo de rede, as duas topologias de árvore e anel são utilizadas, contendo 30 nodos finais conectados em 15 dispositivos de encaminhamento, como apresentadas nas Figuras 2(b) e 2(c). Sobre cada uma dessas redes com topologias diferenciadas, um controlador instala todas as regras de encaminhamento de maneira reativa, ou seja, quando um novo fluxo que não possui uma regra correspondente é identificado pelo dispositivo de encaminhamento, o controlador deve gerar uma nova regra que é instalada na tabela de fluxos.



(a) Topologia em estrela

(b) Topologia em árvore

(c) Topologia em anel

Figura 2. Topologias utilizadas nos experimentos

Carga de trabalho. Para realizar a comparação, é necessário a criação dos fluxos que são trafegados dentro das diferentes topologias de rede propostas. Primeiramente, um par de nodos finais é escolhido aleatoriamente (um cliente e um servidor). Os fluxos são gerados com base em dois parâmetros: duração do fluxo e intervalo entre rajadas. A duração de cada fluxo segue uma distribuição de log-normal com média $\mu = 4s$ e desvio padrão $\lambda = 1s$. O intervalo entre as rajadas de dados é um processo Poisson com média $\lambda = 1s$ [Neves et al. 2016] [Benson et al. 2010]. Fixa-se uma quantidade de 500 fluxos ativos na rede. Esta carga de trabalho representa um cenário realista, onde a maioria dos fluxos possui um ciclo de vida curto, enquanto apenas uma pequena parcela é constituída por fluxos com ciclos de vida longos. Dessa forma, cada experimento realizado, precisa ser executada por no mínimo 12 minutos, pois é o período de tempo necessário para que 500 fluxos sejam iniciados e concluídos.

Métricas. O desempenho de cada forma de gerenciamento espacial é comparado de acordo com quatro métricas: (*i*) quantidade de regras instaladas, (*ii*) quantidade de

intervenções do controlador, (*iii*) latência e (*iv*) *jitter*. Por quantidade de regras instaladas, considera-se a soma das regras presentes nos dispositivos de encaminhamento ao final de cada experimento, sendo a principal métrica que define a eficiência de uma forma de gerenciamento em relação a utilização da capacidade de armazenamento dos dispositivos de encaminhamento. Por quantidade de intervenções do controlador, considera-se a soma das mensagens para inserção de uma nova regra na tabela de fluxos (*flowmods*) e das mensagens de encaminhamento direto de pacotes (*packet-out*), sendo a principal métrica que define a sobrecarga inserida por uma forma de gerenciamento. Por latência, considera-se o tempo que um pacote precisa para alcançar o seu destino e retornar para a origem, também conhecido como *Round Trip Time*. Cada forma de gerenciamento irá influenciar a latência da rede de uma maneira diferente, principalmente, se o tempo de processamento da mesma é alto. Por *jitter*, considera-se a variação do atraso entre os pacotes de dados, sendo a métrica que consegue capturar o impacto das formas de gerenciamento na remoção e instalação de novas regras, gerando momentos instáveis na rede. Para a correta mensuração das métricas de desempenho em termos estatísticos, as coletas foram replicadas no mínimo 20 vezes, alcançando um nível de confiança de 95%.

4.2. Resultados

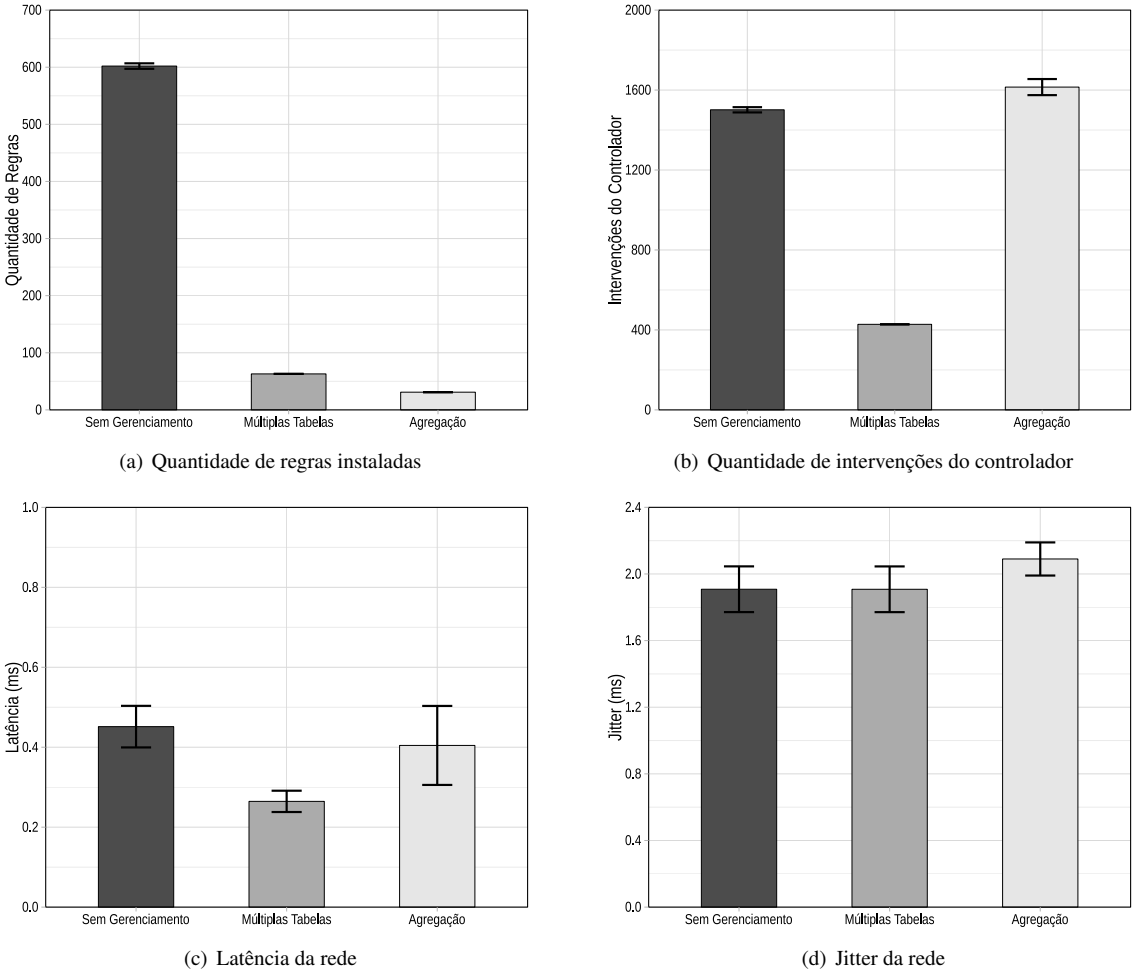
Essa seção apresenta os resultados experimentais obtidos a partir da metodologia proposta, organizados da seguinte forma. Primeiro, analisa-se o comportamento do gerenciamento por agregação e múltiplas tabelas comparando-os para um único dispositivo de encaminhamento na topologia de estrela. Em seguida, realiza-se a comparação entre essas formas de gerenciamento considerando as topologias de árvore e anel. É importante frisar que os experimentos foram realizados em um computador equipado com processador Intel i7-4770S com 4 núcleos de 3.1 GHz e 8GB de RAM. As formas de gerenciamento de domínio espacial foram implementados como aplicações do controlador Ryu (versão 4.2.2). Para mensurar a métrica de *jitter* foi utilizado a ferramenta iPerf versão 2.0.5. Além disso, foram utilizados o Mininet (versão 2.2.1) e Open vSwitch (versão 2.0.2 com suporte a OpenFlow 1.3) para emular uma rede real.

Comparação em uma Topologia Estrela

A topologia estrela permite a comparação entre as formas de gerenciamento de regras de domínio espacial para um único dispositivo de encaminhamento. Resultados coletados a partir do monitoramento desse dispositivo, podem ser observados a partir das Figuras 3(a), 3(b), 3(c) e 3(d). Essas figuras apresentam os gerenciamentos por agregação e múltiplas tabelas, bem como a linha base sem gerenciamento através das colunas e o eixo x.

Na Figura 3(a), o eixo y representa a eficiência de cada uma das formas de gerenciamento a partir da quantidade de regras instaladas no dispositivo de encaminhamento. Nota-se que o gerenciamento por agregação de fluxos reduz significativamente a quantidade total de regras utilizadas, alcançando uma melhora de aproximadamente 95%, quando comparado com a linha base sem gerenciamento. O gerenciamento por múltiplas tabelas, por sua vez, obteve uma redução de aproximadamente 89%, comparado a linha base, ou uma eficiência 6% menor que o gerenciamento por agregação.

Na Figura 3(b), o eixo y representa a quantidade de intervenções do controlador utilizadas para instalar regras e manter o estado da tabela de fluxos ao decorrer do experimentos. O gerenciamento por agregação de fluxos apresentou uma quantidade de



(a) Quantidade de regras instaladas

(b) Quantidade de intervenções do controlador

(c) Latência da rede

(d) Jitter da rede

Figura 3. Resultados obtidos para uma topologia de rede estrela

intervenções do controlador aproximadamente 8% maior do que a linha base e por volta de 370% maior em relação ao gerenciamento por múltiplas tabelas. A grande quantidade de interrupções necessária para a operação do gerenciamento por agregação decorre do processo de atualização da tabela de fluxos com as regras agregadas. Já, o gerenciamento por múltiplas tabelas, obteve uma redução na quantidade de intervenções do controlador de aproximadamente 71% em relação a linha base.

Na Figura 3(c), o eixo y representa a latência média mensurada minuto-a-minuto durante o experimento. Nota-se que o gerenciamento por múltiplas tabelas alcança a melhor latência dentro da rede estrela com 0,26 ms, enquanto o gerenciamento por agregação apresentou 0,40 ms e a linha base 0,45 ms. Os intervalos de confiança permitem identificar que o gerenciamento de múltiplas tabelas é comparativamente inferior aos demais. Entretanto, o mesmo não pode ser afirmado em relação ao gerenciamento por agregação e a linha base, permanecendo como não diferenciáveis para um nível de confiança de 95%.

Na Figura 3(d), o eixo y representa o *jitter* médio coletado durante a duração de cada fluxo. Pode-se observar, que o gerenciamento por múltiplas tabelas e a linha base atingiram um *jitter* médio de 1,9 ms e o gerenciamento por agregação apresenta 2,19 ms. Considerando o intervalo de confiança utilizado, pode-se concluir que as formas de gerenciamento são semelhantes e não diferenciáveis.

Baseado na comparação em uma topologia estrela, pode-se concluir que o gerenciamento por agregação possui a melhor eficiência para a redução das regras de encaminhamento na tabela de fluxos. Entretanto, para atingir essa eficiência são necessárias uma quantidade significativa de intervenções do controlador, inserindo sobrecarga na rede com tráfego de sinalização. Por sua vez, o gerenciamento por múltiplas tabelas, atinge uma eficiência semelhante ao gerenciamento por agregação, mas sem a necessidade de uma grande quantidade de intervenções do controlador, apresentando o melhor custo benefício entre as estratégias de domínio espacial *intra-switch* neste cenário.

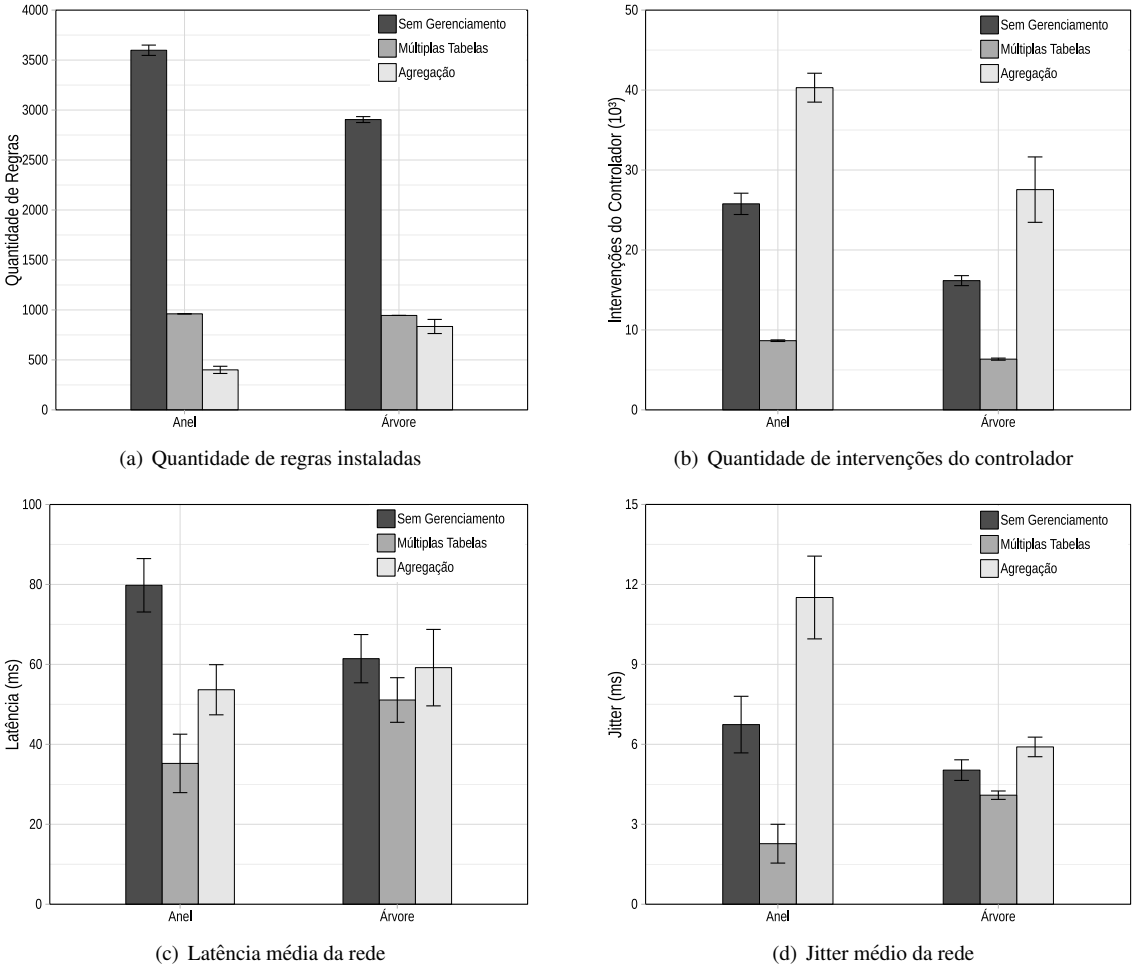
Comparação em Topologias de Árvore e Anel

As redes de topologias em anel e árvore permitem extrapolar a comparação entre as diferentes formas de gerenciamento espacial *intra-switch* para redes com números maiores de dispositivos. Para realizar essa comparação, os resultados coletados a partir do monitoramento dos dispositivo podem ser observados a partir das Figuras 4(a), 4(b), 4(c) e 4(d). Essas figuras apresentam as formas de gerenciamento espacial por agregação e múltiplas tabelas, bem como a linha base sem gerenciamento através das colunas com diferentes cores. Já, no eixo x, as colunas são agrupadas de acordo com as duas topologias.

A Figura 4(a) representa no eixo y a quantidade média das regras instaladas em cada dispositivo de encaminhamento. Para as topologias em anel e árvore, a eficiência do gerenciamento por agregação de fluxos é superior que as demais formas, alcançando uma melhora de 88% para anel e 71% para árvore, quando comparado com a linha base sem gerenciamento. Ainda que seja o mais eficiente, o gerenciamento por agregação apresentou baixa resiliência em relação a troca de topologias, duplicando sua quantidade de regras instaladas entre as topologias de anel para árvore. Já, o gerenciamento por múltiplas tabelas, mostrou-se resiliente a alteração das topologias, mantendo sua eficiência praticamente intacta, com 961 regras instaladas para anel (*i.e.*, uma redução de 73% comparado a linha base) e 945 regras para topologia em árvore (*i.e.*, uma redução de 67% comparado a linha base). É importante salientar que a resiliência é fundamental para a estabilidade e previsão do número de regras a serem utilizadas em redes reprogramáveis.

Na Figura 4(b), o eixo y representa a quantidade de intervenções do controlador ao decorrer do experimentos necessárias para instalar e manter o estado das tabelas de fluxos. Com um intervalo de confiança de 95%, pode-se constatar que o gerenciamento por múltiplas tabelas apresenta a menor quantidade de intervenções do controlador para as topologias de anel (66% abaixo da linha base) e para topologias em árvore (66% abaixo da linha base). Já, o gerenciamento por agregação de fluxo obteve a maior quantidade de intervenções do controlador, 155% acima da linha base para topologia em anel e 170% para topologia em árvore. Um fato interessante é o impacto significante no crescimento da quantidade de intervenções do controlador entre as topologia de árvore para anel, onde, no melhor caso, 2310 intervenções foram acrescidas utilizando o gerenciamento por múltiplas tabelas e, no pior caso, 12460 intervenções extras foram requisitadas pelo gerenciamento por agregação. Assim, a topologia da rede tem grande influência na quantidade de intervenções do controlador sem alteração na quantidade de fluxos ou número de encaminhamento presentes em relação a forma de gerenciamento utilizada.

Na Figura 4(c), o eixo y representa a métrica de latência média da rede. A forma de gerenciamento que propiciou a menor latência média para uma topologia em anel é por



(a) Quantidade de regras instaladas

(b) Quantidade de intervenções do controlador

(c) Latência média da rede

(d) Jitter médio da rede

Figura 4. Resultados obtidos para as topologias anel e árvore

múltiplas tabelas, com uma latência média de 35,22 ms. Já, para a topologia em árvore, todas as formas de gerenciamento, incluindo-se a linha base, são semelhantes dado a intersecção entre as barras de erro calculadas com um nível de confiança de 95%. É importante frisar que o gerenciamento por múltiplas tabelas apresentou uma latência 57% maior entre as topologias de anel para árvore e a mesma análise não pode ser realizada para o gerenciamento por agregação e linha base.

Na Figura 4(d), o eixo y representa o *jitter* médio mensurado nas redes com topologias em anel e árvore. O gerenciamento por múltiplas tabelas possui o menor impacto no *jitter* para ambas as topologias de redes, alcançando o valor médio de 2.27 ms para anel e 4.09 ms para árvore. Já, o gerenciamento por agregação obteve o maior impacto no *jitter* para ambas as redes, apresentando um valor médio de 11,50 ms para anel e 5.9 ms para árvore. É importante observar que as formas de gerenciamento possuem valores semelhantes para ambas as topologias, exceto pelo gerenciamento por agregação com um valor significantemente alto de *jitter*, próximo aos 12 ms. Isto significa, que o atraso da rede gerenciada por agregação de regras torna-se instável e compromete a utilização de aplicações sensíveis ao *jitter*, como *Voice Over IP* e vídeo chats seguros.

Baseado na comparação entre topologias de árvore e anel, percebe-se que o gerenciamento por agregação de fluxos mostrou-se mais eficiente entre as formas de geren-

ciamento espacial *intra-switch*. Entretanto, essa forma de gerenciamento apresentou uma elevada quantidade de intervenções do controlador e degradação da qualidade de serviço da rede. O gerenciamento por múltiplas tabelas demonstrou-se resiliente as topologias mantendo tanto a quantidade de regras instaladas, quanto a número de intervenções do controlador inalterados. Essa forma de gerenciamento não impactou significativamente na qualidade de serviço da rede. Assim, múltiplas tabelas apresenta o melhor custo benefício, para as topologias estrela, anel e árvore.

5. Conclusão e Trabalhos Futuros

Neste artigo apresentou-se uma comparação de duas formas de gerenciamento de regras de domínio espacial *intra-switch*, por agregação e múltiplas tabelas, para redes baseada em OpenFlow. Essas formas de gerenciamento foram comparadas em redes com diferentes topologias, *i.e.,* estrela, anel e árvore. A partir dos resultados obtidos, pode-se afirmar que o gerenciamento por agregação de fluxos reduz significativamente a quantidade de regras de encaminhamento independente da topologia utilizada. Entretanto, essa forma de gerenciamento requer uma quantidade elevada de intervenções do controlador, além de impactar negativamente na latência e no *jitter* das redes. Além disso, o gerenciamento por agregação de fluxos não é resiliente a troca de topologias, tendo sua eficiência alterada. Por outro lado, o gerenciamento por múltiplas tabelas reduz a quantidade de regras de encaminhamento, necessitando uma baixa quantidade de intervenções do controlador e com baixo impacto na latência e *jitter* das redes. Além disso, demonstrou-se resiliente a mudança de topologia, mantendo sua eficiência praticamente constante, tanto para as redes em anel, quanto em árvore. Assim, o gerenciamento por múltiplas tabelas possui o melhor custo benefício, considerando regras de domínio espacial *intra-switch*.

Como trabalhos futuros, pretende-se realizar experimentação em equipamentos reais que utilizem tabela de fluxos implementadas com memórias *Ternary Content-Addressable Memory* (TCAM). Além disso, pretende-se extrapolar a quantidade de estratégias avaliadas realizando um estudo mais abrangente utilizando outras formas de gerenciamento espacial de regras (*i.e.,* posicionamento de regras). Por fim, pretende-se investigar mecanismos que possam melhorar o desempenho das estratégias estudadas oferecendo garantias de desempenho a rede.

Agradecimentos

Este trabalho foi financiado pelo programa Horizon 2020 da União Europeia para pesquisa, desenvolvimento tecnológico e demonstração no âmbito do acordo nº. 688941 (FUTEBOL), bem como pelo Ministério da Ciência, Tecnologia, Inovação e Comunicação (MCTIC) através da RNP/CTIC.

Referências

Benson, T., Akella, A., e Maltz, D. A. (2010). Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pages 267–280, New York. ACM.

Costa, L., Vieira, A., Silva, E., Macedo, D., Gomes, G., Correia, L., e Vieira, L. (2016). Avaliação de desempenho de planos de dados openflow. *34o. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2016*.

Giroire, F., Moulierac, J., e Phan, T. K. (2014). Optimizing rule placement in software-defined networks for energy-aware routing. In *Proc. IEEE Global Communications Conference*, pages 2523–2529.

Kamiyama, N., Takahashi, Y., Ishibashi, K., Shiomoto, K., Otoshi, T., Ohsita, Y., e Murata, M. (2014). Flow aggregation for traffic engineering. In *Proc. IEEE Global Communications Conference*, pages 1936–1941.

Kanizo, Y., Hay, D., e Keslassy, I. (2013). Palette: Distributing tables in software-defined networks. In *Proc. IEEE INFOCOM 2013*, pages 545–549.

Kreutz, D., Ramos, F., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., e Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Lee, B. S., Kanagavelu, R., e Aung, K. M. M. (2013). An efficient flow cache algorithm with improved fairness in software-defined data center networks. In *Proc. IEEE 2nd Int. Conf. Cloud Networking (CloudNet)*, pages 18–24.

Luo, S., Yu, H., e Li, L. M. (2014). Fast incremental flow table aggregation in sdn. In *Proc. 23rd Int. Conf. Computer Communication and Networks (ICCCN)*, pages 1–8.

Neves, M., Oliveira, R., Mazzola, F., Marcon, D., Gaspary, L., e Barcellos, M. (2016). Contando os segundos: Avaliação de estratégias de domínio temporal para a gerência de regras em redes sdn. *34o. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2016*.

Nguyen, X. N., Saucez, D., Barakat, C., e Turletti, T. (2015). Officer: A general optimization framework for openflow rule allocation and endpoint policy enforcement. In *Proc. IEEE Conf. Computer Communications (INFOCOM)*, pages 478–486.

Nguyen, X. N., Saucez, D., Barakat, C., e Turletti, T. (2016). Rules placement problem in openflow networks: A survey. *IEEE Communications Surveys Tutorials*, 18(2):1273–1286.

ONF (2015). The benefits of multiple flow tables and TTPs. Open Networking Foundation. Disponível em: $https://www.opennetworking.org$.

Vishnoi, A., Poddar, R., Mann, V., e Bhattacharya, S. (2014). Effective switch memory management in openflow networks. In *Proc. of the 8th ACM International Conference on Distributed Event-Based Systems*, DEBS '14, pages 177–188, New York. ACM.

Wickboldt, J., De Jesus, W., Isolani, P., Both, C., Rochol, J., e Granville, L. (2015). Software-define networking: Management requirements and challenges. *IEEE Communications Magazine*, 53(1):278–285.

Topologias virtuais confiáveis considerando múltiplos critérios para o contexto borda-como-serviço

Rafael L. Gomes[1,2], **Luiz F. Bittencourt**[2], **Edmundo M. R. Madeira**[2]

[1]Centro de Ciência e Tecnologia (CCT) - Universidade Estadual do Ceará (UECE)

[2]Instituto de Computação (IC) - Universidade Estadual de Campinas (UNICAMP)

{rafaellgom, bit, edmundo}@ic.unicamp.br

Abstract. *Nowadays, the Internet is part of our lives, being an essential way of communication. However, it suffers from limitations that prevent the Internet to guarantee Quality of Service (QoS) to the users. Recently, the scientific community presented the Edge as a Service (EaaS) paradigm as a possible approach to improve the Internet access service. EaaS deploys Virtual Networks (VNs) over Software Defined Networks (SDNs) to ease the management of Internet Service Providers (ISPs). The EaaS needs an algorithm to define virtual topologies for the VNs considering the resource utilization, the energy consumption and the service delivery reliability. Within this context, this paper presents the* Fuzzy for Allocation *(FUZA) algorithm to define reliable virtual topologies based on a fuzzy system considering energy consumption and bandwidth availability. The results suggest that the proposed algorithm can deploy reliable VNs, while improving bandwidth utilization and energy consumption.*

Resumo. *Atualmente a Internet faz parte do cotidiano das pessoas, sendo um meio de comunicação essencial. Contudo, esta sofre de limitações que impedem a Internet de garantir Qualidade de Serviço (QoS) aos usuários. Recentemente, a comunidade científica apresentou o paradigma Borda como Serviço (EaaS) como uma possível abordagem para melhorar o serviço de acesso a Internet. EaaS implanta redes virtuais (VNs) sobre Redes Definidas por Software (SDN) para facilitar o gerenciamento de Provedores de Internet (ISPs). O paradigma EaaS necessita de um algoritmo para gerar topologias virtuais para as VNs considerando a utilização de recursos, consumo de energia e a confiabilidade do serviço prestado. Dentro deste contexto, este artigo apresenta o algoritmo Fuzzy como base para Alocação (FUZA) para definir topologias virtuais confiáveis baseado em um sistema fuzzy que considera o consumo de energia e a disponibilidade de largura de banda. Os resultados sugerem que o algoritmo proposto pode implantar VNs confiáveis, enquanto melhora a utilização de largura de banda e o consumo de energia.*

1. Introdução

A sociedade vem adaptando a abordagem tradicional de comunicação baseada em chamadas de voz e/ou mensagens de texto para chamadas de vídeo em tempo real e/ou compartilhando informações em redes sociais. Porém, os Provedores de Internet (*Internet Service Providers* - ISPs) atuais não conseguem garantir Qualidade de Serviço (*Quality*

of Service - QoS) para este novo aspecto da Internet, o qual gera uma demanda elástica de recursos de rede no decorrer do dia.

Pesquisadores pelo mundo têm investigado formas de lidar com este novo cenário. A Borda como Serviço (*Edge as a Service* - EaaS) é uma abordagem proposta para melhorar a capacidade das redes de borda em prover acesso à Internet [Davy et al. 2014]. EaaS implanta Redes Virtuais (*Virtual Networks* - VNs) sobre Redes Definidas por Software (Software Defined Network - SDN) para proporcionar flexibilidade e gerenciabilidade à alocação de recursos e customização do comportamento da rede.

Tradicionalmente, os ISPs visam maximizar seus lucros, e duas métricas são diretamente relacionada a isso: (i) número de clientes e (ii) consumo de energia. O número de clientes é representado pela quantidade de Acordos de Nível de Serviço (*Service Level Agreements* - SLAs) ativos. Assim, quanto mais clientes, maior é o lucro dos ISPs. ISPs podem aumentar o número de clientes a partir de uma melhor utilização da Largura de Banda (*Bandwidth* - Bw) em suas infraestruturas de rede. Por outro lado, consumo de energia da infraestrutura de rede, bem como a Eficiência Energética (*Energy Efficiency* - EE), vêm se tornando pontos cada vez mais importantes a serem considerados pelos ISPs.

Aplicando a abordagem EaaS, os ISPs precisam atender os parâmetros do SLA, onde confiabilidade é um requisito chave para assegurar a QoS para os usuários. Confiabilidade de rede é vista como a probabilidade da rede manter a comunicação mesmo quando falhas ocorrem na infraestrutura [Lee et al. 2010]. Portanto, confiabilidade engloba não apenas ações reativas para gerenciamento pós-falha, mas também um planejamento estratégico pré-falhas, ou seja, fazer o gerenciamento e implantação dos serviços de forma que a ocorrência de falhas não afete a prestação destes serviços aos clientes.

A implantação de VNs é uma tarefa vital no planejamento estratégico necessário para prover um bom acesso a Internet. Uma das tarefas do processo de implantação de VNs é decidir quais componentes da infraestrutura de rede (enlaces e nós) farão parte de cada VN, ou seja, definir a topologia virtual da VN. Esta decisão deve englobar aspectos como planejamento, confiabilidade, eficiência energética, disponibilidade de Bw, dentre outros. Sendo assim, uma abordagem multicritério é capaz de aprimorar o processo de definição da topologia virtual. A ideia de sistemas Fuzzy é comumente usada como base para tomada de decisão com múltiplos critérios.

Dentro deste contexto, este artigo apresenta o algoritmo Fuzzy como base para Alocação (FUZA), o qual define topologias virtuais em EaaS considerando a EE e a Bw disponível no ISP, enquanto considera a confiabilidade necessária para a VN. O objetivo do algoritmo é definir uma topologia virtual para a VN, onde a topologia é planejada para ser confiável e com a Bw exigida (cumprindo os parâmetros do SLA), bem como, simultaneamente, reduzindo a Bw total alocada e o consumo de energia do ISP.

O desempenho do algoritmo *FUZA* foi avaliado e comparado com alguns algoritmos existentes que podem ser aplicados para o contexto de implantação de VNs: RKSP [Eppstein 1994], MSPS [Parandehgheibi et al. 2014], BRAR [Gomes et al. 2016b] and RDP [Gomes et al. 2016a]. Os experimentos realizados avaliaram os benefícios do algoritmo *FUZA* e dos algoritmos existentes. Os resultados mostram a eficiência do algoritmo *FUZA* em definir VNs confiáveis, enquanto aprimora a EE e a utilização de Bw do ISP.

Este artigo está organizado da seguinte forma: a Seção 2 inclui os conceitos

básicos, ou seja, a apresentação do método de confiabilidade considerado, a descrição do contexto EaaS e os trabalhos existentes mais relacionados; a Seção 3 introduz o algoritmo proposto; a Seção 4 apresenta o resultado dos experimentos realizados; e a Seção 5 conclui o artigo e cita os trabalhos futuros.

2. Contexto

Nesta seção serão detalhados os conceitos chave para a compreensão deste artigo, englobando EaaS, o método de confiabilidade utilizado e alguns trabalhos relacionados ao processo de implantação de VNs em EaaS.

2.1. Edge-as-a-Service (EaaS)

EaaS habilita os provedores de serviço para usar virtualização e construir VNs elásticas [Davy et al. 2014]. EaaS desassocia a relação entre os operadores de rede e suas redes de acesso. Através do desenvolvimento de redes virtuais, pode-se dar suporte a funções adaptativas requisitadas sobre demanda a fim de flexibilizar a entrega de serviços. Em uma abordagem de integração entre SDN e VN, os componentes de hardware podem ser alocados em duas VNs e então acessadas por diferentes controladores de rede. O comportamento de uma VN é definido através da configuração do controlador responsável pela mesma, onde um controlador é responsável por cada VN. A Figura 1 ilustra o cenário EaaS tratado neste artigo, onde duas VNs são implantadas para diferentes clientes. Assim, cada VN é moldada de acordo com as características do cliente.

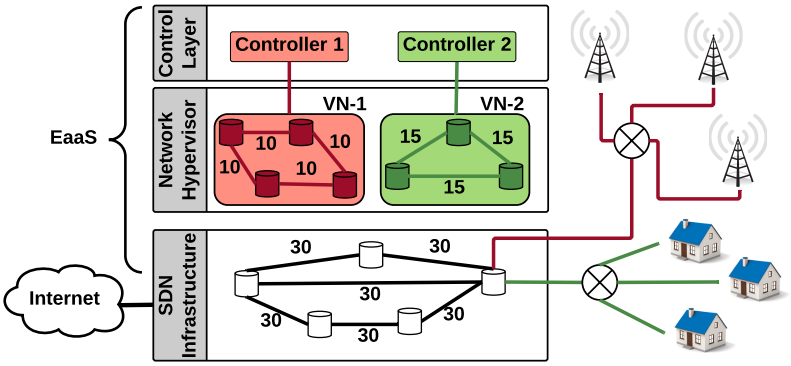


Figura 1. Cenário representando o contexto EaaS.

Os parâmetros da VN englobam algumas métricas, onde o foco deste artigo considera três aspectos: confiabilidade, eficiência energética e largura de banda. A confiabilidade é diretamente relacionada ao número de caminhos arternativos na rede, onde quanto maior é a redundância de caminhos, maior é a confiabilidade da rede. Quando um nó/enlace falha e o caminho primário é afetado, pode-se direcionar o tráfego para o caminho secundário [Lee et al. 2010]. Contudo, não é vantajoso para o ISP definir redundância completa (dois caminhos totalmente disjuntos) para um cliente o qual não deseja tal nível de confiabilidade, visto que isso representa desperdício de recursos.

Um algoritmo é aplicado pelo ISP para definir quais componentes da infraestrutura de rede serão associados a cada VN implantada. Por exemplo, um cliente pode negociar uma VN com um ISP, e este implantar a VN-1 vermelha (Figura 1), que possui Bw de 10 Mbps. O algoritmo proposto neste artigo executa no ISP para implantar as VNs após a especificação do SLA ser finalizada.

Quando parte da rede é comprometida, uma VN pode perder parte dos seus recursos. A definição de uma topologia virtual apropriada pode permitir a operação da VN com uma qualidade mínima definida no SLA através do aumento da Bw alocada no caminho ainda operacional. Dentro deste contexto, o algoritmo *FUZA* proposto neste artigo gera uma topologia virtual confiável que considera tanto consumo de energia quanto a disponibilidade de Bw. Aspectos sobre energia podem ser avaliados sobre diversas perspectivas, neste artigo considerada-se a Eficiência Energética (*Energy Efficiency* - EE) no contexto EaaS, que é definida como o montante de Bw alocada para os clientes em relação a quantidade de energia consumida pela infraestrutura de rede do ISP [Gomes et al. 2016b].

2.2. Confiabilidade de Rede

No contexto de implantação de VNs, inicialmente a topologia virtual é determinada, e posteriormente a confiabilidade da topologia gerada é calculada. Este artigo utiliza a confiabilidade da topologia virtual como restrição de adequabilidade da mesma. O método de medição de confiabilidade é aplicado para avaliar se a topologia virtual gerada atende a confiabilidade mínima requisitada. Desta forma, o método de confiabilidade usado é independente do algoritmo de geração da topologia virtual.

Neste artigo foi usado o método proposto por Li et al. [Li and Silvester 1984], o qual possui as seguintes vantagens: (i) foi proposto para redes genéricas (ou seja, este método não usa informações sobre os dispositivos físicos ou uma base de dados pré-definida); (ii) possui uma boa aproximação da confiabilidade com baixo custo computacional; (iii) é flexível para variar o número de falhas na rede; e (iv) é de fácil adaptação para quando se tem uma fonte e vários destinos.

2.3. Trabalhos Relacionados

Nesta seção serão descritos os trabalhos chave relacionados a definição de topologias virtuais e SDN, os quais consideram a confiabilidade da rede. A Tabela 1 resume os trabalhos descritos, enfatizando a diferença entre eles e a proposta deste artigo. Cada coluna da Tabela 1 especifica uma característica do trabalho: *Contexto* apresenta o ambiente de rede para qual o trabalho foi projetado, e *Critério* é o conjunto de informações nas quais o trabalho se baseia para implantar VNs.

Soualah et al. [Soualah et al. 2014] propõem uma estratégia para mapeamento de redes virtuais dentro de ambientes em nuvem para aprimorar a recuperação quando ocorrem falhas nos equipamentos (enlaces e roteadores). Adicionalmente, a proposta apresentada na referência [Soualah et al. 2014] visa contornar a complexidade exponencial do mapeamento. Contudo, esta abordagem foca em ambientes de nuvem, além de não considerar aspectos de energia.

Cheng et al. [Cheng et al. 2012] apresentam um método para gerar um *ranking* de nós mais adequados e um algoritmo guloso para combinar nós de uma VN para nós na infraestrutura de rede. Da mesma forma, Mano et al. [Mano et al. 2014] descrevem um método para otimizar a implantação de redes virtuais sob múltiplos domínios, sem revelar informações privadas de cada domínio. Apesar de ambas as propostas focarem em implantação de redes virtuais, elas são baseadas em contextos específicos (capacidade de gerar *ranking* e múltiplos domínios) e não consideram aspectos chave deste artigo como consumo de energia, confiabilidade e planejamento estratégico.

Parandehgheibi et al. [Parandehgheibi et al. 2014] propõem um conjunto de algoritmos para lidar com o problema de capacidade de sobrevivência em redes multi camada. Os autores descrevem o Minimum Survivable Path Set (MSPS), o qual é o número mínimo de caminhos necessários para que um par de nós consiga se comunicar caso uma única falha ocorra na infraestrutura de rede. Portanto, o algoritmo MSPS é uma abordagem para trazer resiliência para a rede, sendo uma solução existente a ser comparada com o algoritmo *FUZA* na seção de experimentos.

Uma abordagem popular para lidar com aspectos de confiabilidade em diversos ambientes de rede é a aplicação do algoritmo Reliable K-Shortest Path (KSP) [Eppstein 1994], o qual encontra o menor caminho e mais outros caminhos adicionais até atingir a confiabilidade desejada.

Em propostas anteriores, os autores desenvolveram dois algoritmos para geração de topologias virtuais de acordo com a confiabilidade desejada: o Bandwidth and Reliability According to Redundancy (BRAR) [Gomes et al. 2016b] e o Relative Disjoint Paths (RDP) [Gomes et al. 2016a]. O algoritmo *BRAR* define caminhos alternativos com a melhor relação entre a confiabilidade de rede e a Bw alocada para a VN. O algoritmo *RDP* gera VNs considerando a disponibilidade de Bw nos enlaces e o consumo de energia geral do ISP. Contudo, estes trabalhos não focam simultaneamente na disponibilidade de Bw, eficiência energética e confiabilidade.

Tabela 1. Trabalhos Relacionados

Referência	Contexto	Critério
[Soualah et al. 2014]	Cloud	Falhas
[Cheng et al. 2012]	VN	Rank dos nós
[Mano et al. 2014]	VN	Múltiplos domínios
[Eppstein 1994]	Genérico	Algoritmo KSP
[Parandehgheibi et al. 2014]	Genérico	Sobrevivência
[Gomes et al. 2016b]	VN	Confiabilidade e Bw
[Gomes et al. 2016a]	VN	Energia e Bw
Algoritmo *FUZA* (este trabalho)	EaaS	Confiabilidade, Energia e Bw

3. Algoritmo *FUZA*

Esta seção apresenta o algoritmo *FUZA* para a geração de topologias virtuais confiáveis, o qual é apresentado no Algoritmo 1. O algoritmo *FUZA* é um algoritmo guloso exato, que tem como base a ideia de caminhos relativamente disjuntos, ou seja, é gerado um caminho inicial e posteriormente são adicionados enlaces ao mesmo a fim de alcançar a confiabilidade desejada, enquanto considera-se simultaneamente a eficiência energética e largura de banda disponível para determinar qual redundância de caminhos é a mais adequada. A notação apresentada na Tabela 2 é usada para descrever o algoritmo.

A variável p é a responsável pelo controle da redundância aplicada na topologia virtual. Por exemplo, $p = 0$ representa o caso sem redundância, ou seja, a topologia da rede virtual terá apenas um caminho para cada nó destino. Por outro lado, $p = 1$ é o caso de redundância completa, ou seja, a topologia terá dois caminhos completamente disjuntos para cada nó destino. Da mesma forma, $p = 0.5$ é o caso onde metade dos enlaces do caminho primário serão usados como base para a definição do caminho secundário.

Tabela 2. Notação Utilizada

Símbolo	Descrição
G	grafo representando a infraestrutura de rede
s	nó fonte/raiz que representa o cliente
D	conjunto de destinos o qual deseja-se conectar
L	conjunto de enlaces da infraestrutura de rede
N	conjunto de nós da infraestrutura de rede
l	enlace entre dois nós
w_l	custo/peso do enlace l
$\wp$	quantidade perto de infinito
ε	quantidade próxima a zero
e	número de enlaces a serem atualizados
p	percentual de redundância desejado ($0 \leqslant p \leqslant 1$)
T_1	árvore com os componentes de rede de s para os nós em D
G'	grafo alternativo com w_l atualizados
T_2	árvore com os caminhos alternativos gerados a partir de G'
G_f	topologia final para a rede virtual (junção de T_1 e T_2)
Rr	confiabilidade requisitada pelo cliente
R	confiabilidade da rede virtual G_f
$best$	melhor topologia virtual encontrada até o momento
Bw_l	largura de banda disponível no enlace l
BwO_l	largura de banda original do enlace l
BwR	largura de banda requisitada pelo cliente
E_S	energia gasta por um nó
$E_L(x)$	energia consumida por um enlace quando x Mbps são alocados
En_{Max}	máximo de energia consumida pela infraestrutura de rede
$Score$	nível de adequabilidade da topologia segundo o sistema fuzzy

Inicialmente, o Algoritmo 1 define um laço para iterar entre os possíveis fatores de redundância (p). Dentro do laço, é criada uma árvore inicial T_1 com o nó s como raiz executando-se o algoritmo de definição de caminho (função $PathDefinition(Grafo, No)$). Esta função encontra o melhor caminho para os nós determinados (em D) de acordo com o peso dos enlaces.

Portanto, T_1 contém os enlaces pertencentes aos caminhos de s para os nós contidos em D. A linha 4 atribui a e o número de enlaces a serem atualizados para gerar a redundância na topologia. e é calculado como um percentual do número de enlaces em T_1 de acordo com p ($0 \leqslant p \leqslant 1$).

Após executar a função $PathDefinition$, o Algoritmo 1 atualiza o custo de cada enlace na rede, criando um novo grafo G'. Para criar este novo grafo, o algoritmo substitui o peso w_l dos enlaces de acordo com a redundância, ou seja, ele substitui o peso w_l dos e primeiros enlaces de T_1 de acordo com a função $UpdateLink(Link, Link)$ apresentada no Algoritmo 2.

No próximo passo, o algoritmo encontra a árvore T_2 no grafo G' (o qual possui os pesos dos enlaces atualizados) a partir da execução da função $PathDefinition$. Pos-

Algoritmo 1 Fuzzy como base para Alocação (FUZA)

1: $p = 0$; ▷ Caso sem redundância
2: **enquanto** $p \leq 1$ **faça** ▷ Redundância $\neq$ Completa
3: Árvore $T_1 = PathDefinition(G,\ s,\ D)$;
4: $e = p * |T_1|$;
5: **para todo** Enlace $j \in T_1$ **faça**
6: **para todo** Enlace $i \in G$ **faça**
7: $UpdateLink(j, i)$;
8: **fim para**
9: **fim para**
10: Árvore $T_2 = PathDefinition(G',\ s,\ D)$;
11: Grafo $G_f = MergePaths(T_1,\ T_2)$;
12: Score $= Fuzzy(Bw_{Impact}(G_f), En_{Impact}(G_f))$;
13: **se** $(Score < best)$ **então**
14: $R = Reliability(G_f)$;
15: **se** $(R > Rr)$ **então**
16: $best = Score$;
17: **fim se**
18: **fim se**
19: $Incrementar(p)$;
20: **fim enquanto**

teriormente, o algoritmo faz a junção das árvores T_1 e T_2 (linha 11) para criar um grafo com os componentes de rede (enlaces e nós) que existem nas duas árvores, resultando na topologia final G_f.

Após a definição de G_f, o algoritmo verifica se esta é a melhor solução encontrada até o momento, baseado no sistema fuzzy a ser descrito na Seção 3.1, sobre as perspectivas de largura de banda disponível (função $Bw_{Impact}(Grafo)$) e consumo de energia (função $En_{Impact}(Grafo)$), ambas descritas nos Algoritmos 3 e 4, respectivamente. $best$ é uma variável para identificar a topologia virtual mais adequada. Se a "melhor" opção é encontrada, checa-se se G_f atende a confiabilidade requisitada pelo cliente no SLA (Rr). A confiabilidade R de G_f é calculada de acordo com o método mostrado na Seção 2.2.

Finalizando o laço, a redundância desejada é incrementada para permitir que menos enlaces que foram utilizados no caminho primário sejam considerados na busca por um caminho alternativo. O valor do incremento é configurado pelo administrador da rede, por exemplo: 0.1, 0.25, ou 0.5 por iteração. Nos experimentos realizados, considerou-se um incremento de 0.25, visto que ele representa uma maior abrangência das possíveis variações de topologia. Portanto, este valor de 0.25 resulta em um número plausível de possibilidades, enquanto que evita uma busca excessiva por topologias com uma pequena variação (muito similares).

O Algoritmo 2 é utilizado para realizar a atualização do custo dos enlaces, o qual tem por objetivo evitar que os enlaces já alocados sejam evitados na busca por um caminho alternativo. Enquanto o número de enlaces (e) não é alcançado, o custo do enlace é substituído por $\wp$ para evitar o seu uso, e após e enlaces serem processados, o custo dos enlaces é substituído por 0, a fim de encorajar a alocação do mesmo.

Algoritmo 2 $UpdateLink(Enlace\ \ j,\ Enlace\ \ i)$

```
se (i == j) então
    se (e > 0) então
        w'_j = ℘ ;
        e = e − 1 ;
    senão
        w'_j = 0;
    fim se
fim se
```

Com relação ao impacto da alocação de G_f na largura de banda, este cálculo é feito de acordo com o Algoritmo 3, onde Bw_l é a largura de banda disponível no enlace l e BwR é a largura de banda requisitada pelo cliente no SLA.

Algoritmo 3 $Bw_{Impact}(Grafo\ \ G_f)$

```
Sum = 0;
para todo Enlace l ∈ G_f faça
    Sum += BwR / Bw_l ;
fim para
retorne Sum;
```

O impacto da alocação de G_f no consumo de energia é medido de acordo com o Algoritmo 4. Primeiramente, a função calcula o consumo de energia dos comutadores (do inglês *switch*) da rede, onde $E_S(Node)$ é a energia gasta pelo nó de acordo com a Equação (1) [Mahadevan et al. 2009], e En_{Max} representa o máximo de energia consumida pela rede (ou seja, a energia consumida quando todos os componentes estão operacionais em máxima capacidade).

$$E_S(o) = P_{ch} + (N_o * Pl) + \sum_{c=0}^{C}(NumPorts_c * E_c) \tag{1}$$

P_{ch} é a energia consumida pelo chassi do *switch*; Pl é a energia consumida pelas portas de transmissão não ativadas, e N_o é o número de placas de transmissão conectadas ao *switch* o. A variável C representa as possíveis configurações de velocidade das portas de transmissão, e E_c é a energia consumida por cada porta executando a uma velocidade c; $NumPorts_c$ é o número de portas configuradas para a velocidade c. De acordo com a referência [Mahadevan et al. 2009], P_{ch} é normalmente 50 Watts e Pl é 40 watts.

Adicionalmente, o Algoritmo 4 calcula o aumento na energia consumida nos enlaces devido ao aumento na alocação de largura de banda requisitada (BwR) no mesmo. BwO_l é a largura de banda original do enlace l, enquanto que $E_c(x)$ é a largura de banda consumida quando x Mbps são alocados no enlace (seguindo a Equação 2, a qual é baseada na referência [Mahadevan et al. 2009]).

$$E_c(x) = \begin{cases} 0.4,\ se\ 0\ < x\ \leq\ 10Mbps; \\ 0.5,\ se\ 10\ < x\ \leq\ 100Mbps; \\ 1\ se\ 100\ < x\ \leq\ 1Gbps; \\ 0,\ caso\ contrario; \end{cases} \tag{2}$$

Algoritmo 4 $En_{Impact}(Graph\ G_f)$

1: $Sum = 0$;
2: **para todo** Nodes $o \in G_f$ **faça**
3: $\quad Sum += \frac{E_S(o)}{En_{Max}}$;
4: **fim para**
5: **para todo** Link $l \in G_f$ **faça**
6: $\quad Sum += \frac{E_C(BwO_l + BwR - Bw_l) - E_C(BwO_l - Bw_l)}{En_{Max}}$;
7: **fim para**
8: retorne Sum;

Numa visão geral, o Algoritmo 1 constrói uma árvore inicial e, a partir de uma atualização dos pesos dos enlaces, busca-se por um caminho alternativo através da adição de novos enlaces a árvore inicial. Esta adição é limitada pelo fator de redundância definido. A atualização do custo dos enlaces é usada para evitar o uso dos enlaces que já foram alocados, mas sem descartá-los como uma opção, forçando o algoritmo a procurar caminhos alternativos para alcançar os nós em D. Executando o Algoritmo 1 gera-se uma topologia para VN com caminhos alternativos considerando aspectos de consumo de energia, largura de banda disponível e a confiabilidade requisitada pelo cliente.

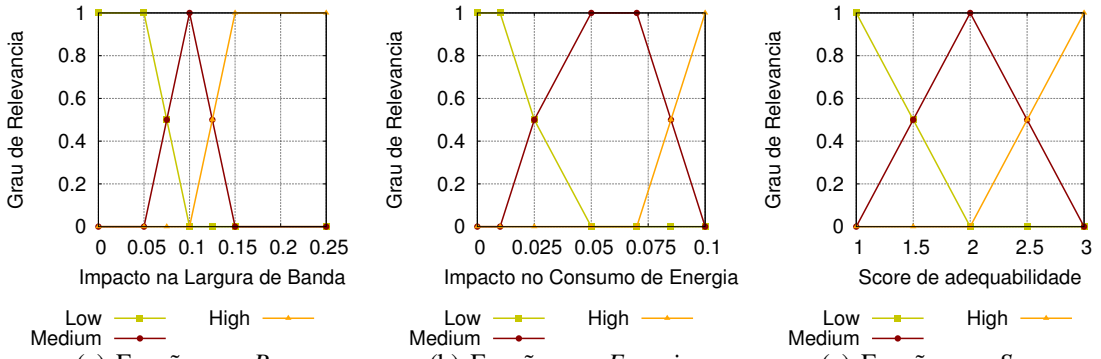
3.1. Sistema fuzzy para medição de adequabilidade

No conceito tradicional de conjuntos, um elemento pertence totalmente ou não a um certo conjunto. Por outro lado, os conjuntos *Fuzzy* definem funções de adesão limitadas ao intervalo $[0, 1]$, para expressar o grau de adesão de um elemento em relação ao conjunto em questão, representado por $\mu(x)$ [Abedin et al. 2011]. Além disso, os conjuntos *Fuzzy* são uma ferramenta para lidar com problemas de tomada de decisão, devido a duas razões: (i) lidam com o conceito de "grau de satisfação"; e (ii) têm uma estrutura matemática para manipular informações vagas.

Inicialmente, os valores de consumo de energia (En_{Impact}) e largura de banda (Bw_{Impact}) são convertidos em variáveis linguísticas, a partir das funções de adesão respectivas, ilustradas nas Figuras 2(a) e 2(b). Posteriormente, as variáveis linguísticas geradas são aplicadas no conjunto de regras (apresentado na Tabela 3) e produzem um conjunto de variáveis linguísticas relacionadas a função de adesão *Score*, a qual é usada como saída e é mostrada na Figura 2(c). A configuração definida do conjunto de regras visa favorecer o uso de caminhos com pequeno impacto na Bw disponível (valores *Low* e *Medium*) e que consigam, ao menos, manter o consumo de energia atual do ISP (valor *Low*).

Por fim, as variáveis linguísticas vindas do sistema de inferência são convertidas em um valor real de acordo com o método Weight Average Maximum (Equação 3), pois é um método eficaz e de baixo processamento, encaixando-se no escopo do trabalho que visa realizar as alocações de VN em tempo real. Este método produz um valor numérico considerando o peso médio dos maiores valores (pico das variáveis linguísticas mostradas na Figura 2(c)).

$$Score = \frac{(1 * \mu_H(x)) + (2 * \mu_M(x)) + (3 * \mu_L(x))}{(\mu_H(x) + \mu_M(x) + \mu_L(x))} \tag{3}$$



(a) Função para *Bw*. (b) Função para *Energia*. (c) Função para *Score*.

Figura 2. Funções de Adesão

Tabela 3. Conjunto de Regras

Energia	Operação	Largura de Banda	*Score*
High	E	High	Low
High	E	Medium	Low
High	E	Low	Low
Medium	E	High	Low
Medium	E	Medium	Medium
Medium	E	Low	Medium
Low	E	High	Medium
Low	E	Medium	High
Low	E	Low	High

Portanto, o resultado oriundo da Equação 3 representa o $Score$ associado a topologia de rede gerada pelo Algoritmo 1. Na Equação 3, μH(x) é o grau de adesão da variável *High*, μM(x) é o grau de adesão da variável *Medium*, e μL(x) é o grau de adesão da variável *Low*. Os valores 1, 2, e 3 são os máximos das variáveis *High*, *Medium*, e *Low*, respectivamente, como mostrado na Figura 2(c).

4. Resultados

Esta seção apresenta os experimentos realizados para avaliar o algoritmo *FUZA* com relação ao processo de definição de topologias virtuais. A fim de realizar os experimentos foi desenvolvido um simulador[1] de alocação de redes virtuais, o qual possibilita a análise dos principais aspectos relacionados a definição de topologias virtuais: alocações realizadas com sucesso, eficiência energética, disponibilidade de largura de banda e a conectividade das redes virtuais quando ocorrem falhas na infraestrutura de rede (onde a ordem das falhas foi gerada de forma aleatória). A Seção 4.1 descreve o cenário usado na avaliação, enquanto a Seção 4.2 discute os resultados e analisa o desempenho dos algoritmos avaliados.

4.1. Configuração do Cenário

Os experimentos utilizaram a topologia da rede GEANT, a qual possui quarenta nós e sessenta enlaces, como ilustrado na Figura 3. Cada enlace da rede GEANT foi configurado com 1Gpbs de largura de banda disponível.

[1]http://bitbucket.org/rafaellgom/vn-allocation/

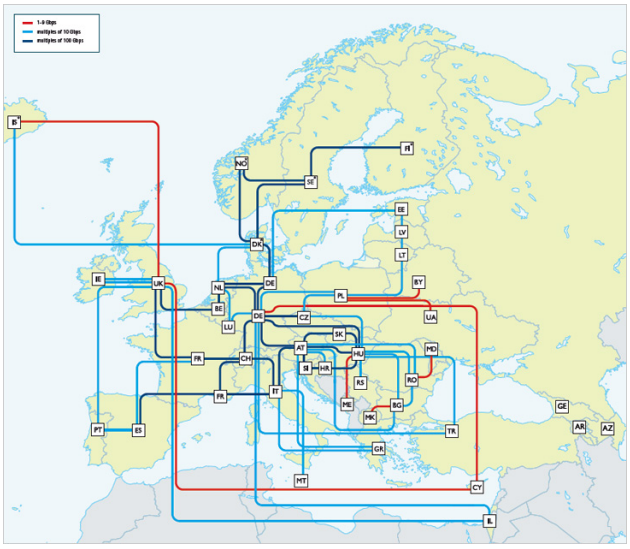


Figura 3. Topologia da rede GEANT.

Os experimentos visam avaliar a capacidade da proposta solucionar um conjunto de requisições de VN com os seguintes parâmetros: (i) conjunto de nós a serem conectados (escolhidos de forma uniforme); (ii) duração das requisições (valor médio de 50 unidades de tempo); e, (iii) Bw requisitada (valor médio de 100 Mbps). Foram gerados aleatoriamente quinhentos conjuntos, onde cada conjunto era composto de cem requisições de redes virtuais.

Normalmente, os modelos de tráfego de rede assumem que o intervalo de tempo de chegada e a duração dos fluxos, os quais resultam na demanda de tráfego, seguem uma distribuição exponencial [Chen 2007]. Portanto, neste trabalho a largura de banda requisitada e a duração das requisições de VN são geradas a partir de um processo Poisson, visto que ambos podem ser considerados relacionados a demanda de tráfego.

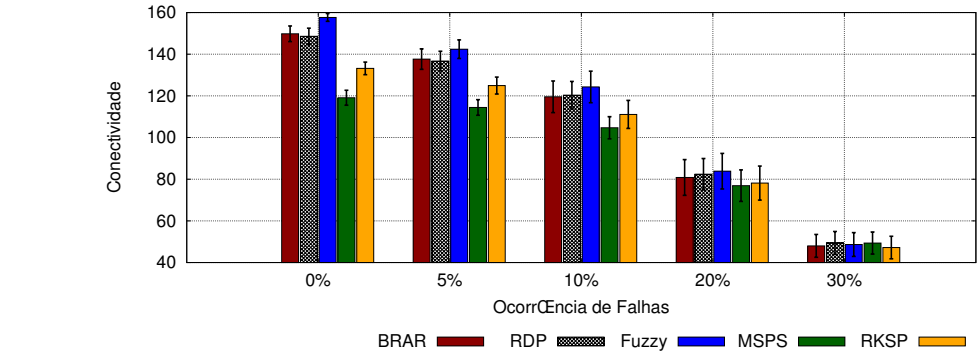
4.2. Resultados

Nesta seção apresenta-se o resultado dos experimentos realizados, comparando o desempenho do algoritmo *FUZA* com os algoritmos existentes descritos na Seção 2.3: RKSP [Eppstein 1994], o qual encontra os K caminhos mais curtos até atingir a confiabilidade desejada; MSPS [Parandehgheibi et al. 2014],um algoritmo que encontra o menor número de caminhos entre dois pares de nós que sobrevivem a qualquer falha de enlace na infraestrutura de rede; BRAR [Gomes et al. 2016b], que busca pela melhor relação confiabilidade e largura de banda; e RDP [Gomes et al. 2016a], um algoritmo de alocação para gerar VN considerando a confiabilidade mínima necessária, assim como a disponibilidade de largura de banda e consumo de energia.

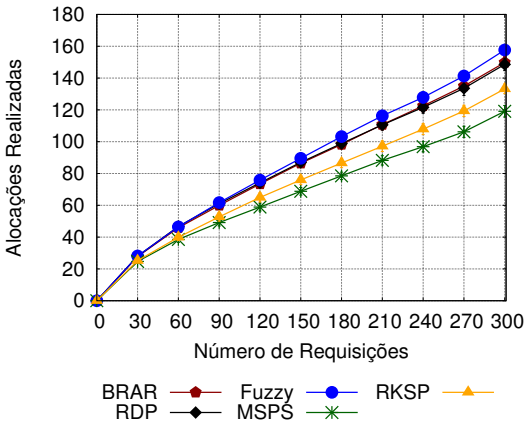
Durante os experimentos foram avaliadas quatro métricas:

- Conectividade pós-falha (Figura 4(a)): o número de requisições bem sucedidas que continuam totalmente conectadas (alcançam todos os nós) após a falha dos componentes de rede;
- Alocações bem sucedidas (Figura 4(b)): o número de requisições solucionadas, ou seja, quantas requisições de VN o algoritmo foi capaz de alocar com a confiabilidade e a largura de banda desejadas;

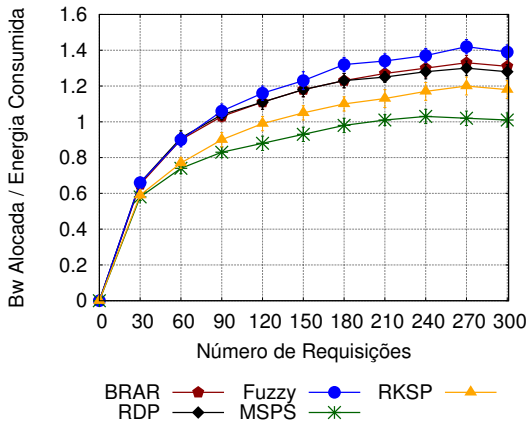
- Eficiência energética (Figura 4(c)): montante de largura de banda alocada em relação ao consumo de energia consumida;
- Largura de banda disponível (Figura 4(d)): a largura de banda média disponível na infraestrutura de rede após as alocações serem realizadas;
- Enlaces saturados (Figura 4(e)): o número de enlaces que ficaram com menos de 10% de largura de banda disponível.



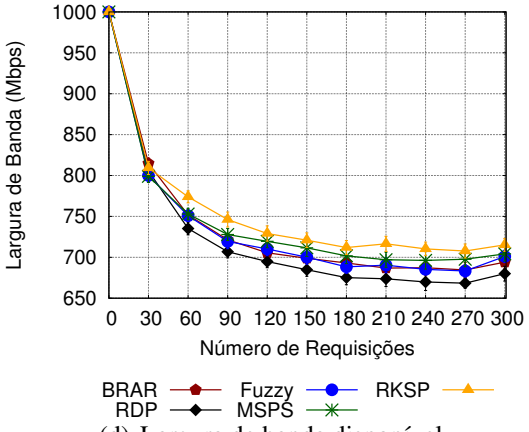
(a) Conectividade pós-falha.



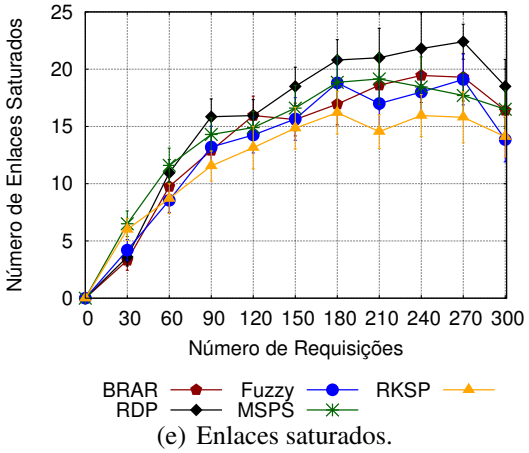
(b) Alocações bem sucedidas.



(c) Eficiência energética.



(d) Largura de banda disponível.



(e) Enlaces saturados.

Figura 4. Desempenho dos algoritmos avaliados.

A Figura 4(a) mostra o número de alocações de VN que continuam operacionais (mantêm conectividade entre todos os nós envolvidos) após ocorrerem falhas. O eixo x da

Figura 4(a) representa o percentual de componentes de rede que aleatoriamente falharam, onde a falha de um enlace representa a remoção do mesmo da rede, enquanto a falha de um nó significa a remoção do nó e seus enlaces da rede. O nível de conectividade é diretamente relacionado ao número de componentes alocados pelo algoritmo para a VN, visto que quanto maior o número de componentes na VN, maior é a probabilidade de ocorrer uma quebra na conectividade quando falhas ocorrem.

O ponto "0%" da Figura 4(a), assim como na Figura 4(b), ilustra o número de requisições alocadas com sucesso (isto é, cumprem o SLA) quando a rede está totalmente operacional. Em geral, o algoritmo *FUZA* solucionou em média, pelo menos, 5% mais requisições que os demais algoritmos. Este melhor desempenho ocorre devido a característica do algoritmo *FUZA* em considerar tanto os aspectos de energia quanto a largura de banda, resultando em um melhor balanceamento de alocações por toda a infraestrutura de rede.

A Figura 4(c) mostra a eficiência energética, a qual avalia se o consumo de energia do ISP está sendo eficaz (quanto maior melhor). Assim como apresentado na Seção 3, a eficiência energética é a quantidade de Bw alocada para o conjunto de VNs pelo montante de energia gasto pela infraestrutura de rede. De acordo com a Figura 4(c), o algoritmo *FUZA* melhora a eficiência energética da rede em torno de 7%. Este melhor desempenho do algoritmo *FUZA*, em relação aos algoritmos existentes, deve-se a abordagem de avaliação considerando ambos os aspectos, de energia e Bw.

De acordo com a Figura 4(d), os algoritmos MSPS, RKSP, e BRAR obtêm um comportamento similar, onde o comportamento dos algoritmos MSPS e RKSP justifica-se pelo baixo número de alocações. Por outro lado, o algoritmo *FUZA* apresenta uma disponibilidade similar aos demais, mesmo realizando mais alocações de VNs. Adicionalmente, quando analisa-se os dados ilustrados na Figura 4(e), percebe-se que os algoritmos possuem um número próximo de enlaces saturados, apesar do algoritmo *FUZA* resultar em um maior número de alocações.

Baseado nos experimentos realizados, os resultados indicam que o algoritmo *FUZA* pode evoluir o processo de alocação de VNs para os ISPs. O algoritmo *FUZA*, apesar de ter um consumo de disponibilidade de Bw similar aos demais algoritmos, consegue alocar mais requisições (em torno de 5% maior) e eleva a eficiência energética do ISP (em torno de 7%). Da mesma forma, o algoritmo *FUZA* aloca VNs que mantêm a conectividade quando falhas na infraestrutura de rede ocorrem.

5. Conclusão

A aplicação da abordagem EaaS traz flexibilidade e gerenciabilidade para os ISPs, permitindo melhorar o uso da infraestrutura de rede e a prestação de serviço aos seus clientes. Contudo, a alocação de VNs ainda é um desafio em aberto. Portanto, este artigo propõe um algoritmo de definição de topologias virtuais multicritério, chamado de *FUZA*.

O foco do algoritmo *FUZA* é definir topologias virtuais para VNs, as quais são planejadas para serem confiáveis em frente a falhas (cumprindo os parametros do SLA), enquanto reduz a utilização de Bw e consumo de energia quando a rede está totalmente operacional. O algoritmo *FUZA* supera os algoritmos existentes analisados, atendendo um maior número de clientes (em torno de %5) e obtendo uma maior eficiência energética (cerca de 7%). Estes fatos habilitam a maximização dos lucros do ISP.

Como trabalhos futuros, pretende-se estender o algoritmo proposto para aplicar uma abordagem multicritério ótima, como por exemplo fronteira de Pareto, bem como adicionar critérios adicionais ao sistema fuzzy.

Agradecimentos

Os autores gostariam de agradecer a FAPESP, CAPES e CNPq pelo apoio financeiro.

References

Abedin, F., Chao, K.-M., and Godwin, N. (2011). A fuzzy group decision making process in a multi-agent negotiation environment. In *15th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 311–318.

Chen, T. M. (2007). Network Traffic Modeling. pages 326–339.

Cheng, X., Su, S., Zhang, Z., Shuang, K., Yang, F., Luo, Y., and Wang, J. (2012). Virtual network embedding through topology awareness and optimization. *Computer Networks*, 56(6):1797 – 1813.

Davy, S., Famaey, J., Serrat-Fernandez, J., Gorricho, J., Miron, A., Dramitinos, M., Neves, P., Latre, S., and Goshen, E. (2014). Challenges to support edge-as-a-service. *Communications Magazine, IEEE*, 52(1):132–139.

Eppstein, D. (1994). Finding the k shortest paths. In *35th Annual Symposium on Foundations of Computer Science*, pages 154–165. IEEE.

Gomes, R. L., Bittencourt, L. F., Madeira, E., Cerqueira, E., and Gerla, M. (2016a). State-Aware allocation of reliable virtual software defined networks based on bandwidth and energy. In *13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 418–423, Las Vegas, USA.

Gomes, R. L., Bittencourt, L. F., Madeira, E. R., Cerqueira, E., and Gerla, M. (2016b). A combined energy-bandwidth approach to allocate resilient virtual software defined networks. *Journal of Network and Computer Applications*, 69:98 – 106.

Lee, H.-W., Modiano, E., and Lee, K. (2010). Diverse Routing in Networks With Probabilistic Failures. *IEEE/ACM Transactions on Networking*, 18(6).

Li, V. and Silvester, J. (1984). Performance Analysis of Networks with Unreliable Components. *IEEE Transactions on Communications*, 32(10):1105–1110.

Mahadevan, P., Sharma, P., Banerjee, S., and Ranganathan, P. (2009). Energy Aware Network Operations. In *IEEE INFOCOM Workshops 2009*, pages 1–6.

Mano, T., Inoue, T., Ikarashi, D., Hamada, K., Mizutani, K., and Akashi, O. (2014). Efficient virtual network optimization across multiple domains without revealing private information. In *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–8.

Parandehgheibi, M., Lee, H.-W., and Modiano, E. (2014). Survivable path sets: A new approach to survivability in multilayer networks. *Journal of Lightwave Technology*, 32(24):4741–4752.

Soualah, O., Fajjari, I., Aitsaadi, N., and Mellouk, A. (2014). A reliable virtual network embedding algorithm based on game theory within cloud's backbone. In *2014 IEEE International Conference on Communications (ICC)*, pages 2975–2981.

Trilha Principal do SBRC 2017
Sessão Técnica 24
Segurança em Redes II

Towards effective reproducible botnet detection methods through scientific workflow management systems

Frederico Tosta de Oliveira[1], **Maria Claudia Cavalcanti**[1], **Ronaldo Moreira Salles**[1]

[1]Instituto Militar de Engenharia (IME)
Praca General Tiburcio, 80 – 22290-270 – Rio de Janeiro – RJ – Brasil

{tosta,yoko,salles}@ime.eb.br

Abstract. *Even after nearly two decades of the creation of the first botnet, the detection and mitigation of their attacks remain one of the biggest challenges faced by researchers and cyber-security professionals. Although there are numerous studies related to botnet detection, estimating how much one method is better than another is still an open problem, mainly because of the difficulty in comparing and reproducing such methods. This work proposes an architecture, implemented with Spark as a high-performance data processing solution, together with VisTrails as a workflow management and data provenance solution, to address this comparability and reproducibility problem in a large-scale environment, as well as a tool, ProvTracker, to analyze and compare the methods results. Another contribution is on the way to couple these two technologies so that intermediary data is maintained available during several partial (re)executions of the experiments involved in each method, minimizing the impact on the analysis of the large amount of data involved.*

1. Introduction

In the past, the malware was limited to cause local impacts, but with the creation of the internet, a new opportunity raised for those malicious software, which spread to millions of computers around the world. At the same time, the ”hijacking” of computers started, when computers are controlled remotely by a new family of malware called bot. This network of compromised machines, botnet, is then controlled by one or more people, botmasters [Hogben et al. 2015].

There are several methods and techniques to detect botnets [Silva et al. 2013, Garcia et al. 2014b]. Among them, we can highlight the methods based on passive network monitoring, where the most prevalent is the detection based on machine learning techniques, e.g. Decision Tree and Random Forest [Biglar Beigi et al. 2014]. In this technique, a set of features contained in the network flows [Hofstede et al. 2014] is handled by the learning algorithm to determine which flow can be considered malicious, i.e., results from the communication between the botmaster and bots.

Currently, experiments for evaluating botnet detection methods are done in an *ad-hoc* way, making it difficult to register valuable information of all the steps involved in the experiment, such as dataset, programs and algorithms used, as well as which parameters' values were tested, which features of the network flows were used, results achieved, metrics, etc.

Even though most of the activities involved in machine learning technique are commonly used in several approaches, as far as we investigated, there is no tool or envi-

ronment able to support the entire life cycle of the botnet detection experiment, allowing the researchers to create, execute, and analyze their experiment.

This gap makes it almost impossible to reproduce and compare these experiments' results and, consequently, makes it difficult to select a detection method to deploy in a real production environment. This keeps the solutions restricted to the study environment of their creators, offering no real alternative to the botnets dismantling. In addition, a botnet experiment involves the processing of high volume of data.

There are generic approaches already proposed, which can be useful to support botnet detection experiments. For instance, there are initiatives on managing workflows [Shi Meilin et al. 1998] and provenance capture [Freire et al. 2008]. In addition, there are also initiatives on using high performance platform for provenance capture [Korolev and Joshi 2014, Akoush et al. 2013], which we will highlight in the following sections. However, they are not properly tailored to address the problems mentioned before.

This work specifies and implements an architecture to address botnet detection experiments. The main paper contribution is on the innovative way it combines the functionality of workflow management systems (WfMS), provenance capture and high performance solutions. Specifically, it provides an infrastructure to support the botnet experiments requirements: creation, execution and analysis. It is implemented using VisTrails WfMS and Spark, together with our tool, ProvTracker. Above all, researchers are able to share their experiments, allowing others to reproduce and compare results. After concluding the experiments, the best approach may be used in a production environment.

The remainder of this paper is structured as follows. Next section presents the related works. Section 3 introduces some concepts for better understanding the proposed architecture, which is presented in Section 4. Section 5 describes the architecture implementation and the technologies involved. Sections 6 and 7 present, respectively, some experiments' results and a conclusion pointing to future work.

2. Related Work

In [Aviv and Haeberlen 2011], the authors explore the challenges to carry out the evaluation of botnet detection systems. Many of the challenges are related to the difficulty in obtaining and sharing a real and diverse network traffic dataset. This is justified by the privacy of the data on the traffic, but ends up impacting directly on the inability to perform qualitative comparisons and repeat third party experiments. They also emphasize the lack of methodology to drive botnet detection experiments.

In [Garcia et al. 2014a], the authors present an empirical comparison of three different botnet detection methods. They point out that the results achieved by botnet detection experiments usually do not compare to other methods. Among the existing factors that prevent them to perform the comparison, they highlight the difficulty of obtaining or sharing datasets, lack of an appropriate description of the experiments, lack of comparison methods and error metric. They propose a new metric errors and conclude that a joint effort to create an execution and comparison platform to botnet detection methods can dramatically improve the results achieved in the area.

In [Garcia et al. 2014b], the authors present a review on botnet detection methods.

They compare and classify fourteen botnet detection methods and highlight that ten could not be reproduced by lack of information about the experiment and twelve could not be reproduced because the datasets were not published. They enumerate a list of desired properties in every publication concerning the detection of botnet, where we can highlight: reproducibility; experiment setup; metrics; and comparison of results. They also speak of the possibility of creating a hybrid detection method, and performing various algorithms simultaneously using a big data solution.

In works such as [Singh et al. 2014, Saad et al. 2011, Biglar Beigi et al. 2014, Zhao et al. 2012, Zhao et al. 2013], the authors implement botnet detection methods using various machine learning techniques. They talk about the importance of continuous training and parameters refinement of the learning algorithms to deal with the botnet mutable behavior in the network traffic. Something that requires a well-documented history of the experiments. [Singh et al. 2014] also raises the need of a big data solution to handle all the network traffic volume.

All the works presented exposed, in some way, the need for a methodology or tool to perform botnet detection experiments so they can be reproduced and compared as well as allow them to evolve their methods in a systematic manner, either by reconfiguration the detection algorithms, either by the selection of new network flows features. Moreover, some authors also mentioned that network flows are difficult to process given its real time and big volume characteristics.

In summary, for botnet detection experiments, it is necessary to address the following requirements: support for creation, execution and reproduction of the experiments, comparability and analysis of their results, and ability to deal with big volume data. However, none of those works propose a tool or approach in this direction. On the other hand, there are generic approaches, already proposed, which can be useful, if combined with each other. These initiatives are discussed in the next section.

3. Background

Typically, a large scale scientific experiment consists of three basic activities : *capture*, *curation* and *analysis*. The *capture* is the acquisition of data, which appear in various shapes and scales, from sensors, simulations, etc. The *curation* may involve activities such as filtering the data and finding the correct data structure to store them. The data *analysis* covers several tasks during the activities flow, such as the use of databases, the application of data mining algorithms, modeling and visualization [Hey et al. 2009].

These kinds of experiments are usually composed by chaining a set of tasks that handle a huge amount of data. Typically, these experiments are created manually by connecting programs inputs and outputs, producing an execution flow plan [Davidson and Freire 2008]. This plan is also known as a workflow. In addition, each plan is configured through specific parameters and input settings. Often, after its execution, the outputs are analyzed and, depending on its performance, this may incur on changes on the parameter settings, aiming at better results. According to [Hey et al. 2009], the workflow is becoming a paradigm to large scale science, through the management of data preparation (capture and curation) and their analysis.

Workflow Management Systems (WfMS), which are automated coordination en-

gines, are used to specify, instantiate, execute, audit and develop a workflow [1]. The WfMS often provides [Taylor et al. 2007]: a visual scripting interface so that scientists can design these workflows (plans) without programming; a way to integrate and access computing platforms transparently; means to conduct analyzes (workflows executions) over various sets of data in a systematic and automatic manner; a way to capture the configuration of the programs involved in these executions (workflow configuration) so that the results can be reproduced and the methods can be reviewed, validated, repeated and adapted. Moreover, WfMS optimize the use of computing resources, allowing partial execution of workflows, as well as rerunning tasks that have failed without the need to rerun the entire workflow [Gil et al. 2007].

Several authors say that the artifacts (data and programs) involved in scientific experiments should contain information about their origins. This information should include items such as who and what process produced the artifact, which sources were used, which parameters and settings were used during the experiment, when the experiment or part of it was performed, among others. All this information related to the artifact origins is also known as *provenance*. Provenance is also important to workflows and scientific experiments [Freire et al. 2008, Moreau et al. 2011]. It can be used as an aid to evaluate the quality and reliability of experiments, since it allows to interpret and understand their results, to check the sequence of steps that led to these results, to identify the entries of these experiments and, if possible, to reproduce their results [Freire et al. 2008].

Some authors proposed approaches to handle provenance and reproducibility in a big data environment: In [Korolev and Joshi 2014], the authors propose a tool to track provenance and allow the reproduction of experiments involving Big Data workflows using an instrumented version of the PIG language. The proposed tool is based on Git, Git-Annex and Git2Prov. In order to maintain the provenance of the data, the system needs to connect to the repository and register the source through Git2Prov. Although the system provides a high degree of reproducibility of the experiment, the author does not mention the overhead introduced by successive synchronizations with Git. The need for synchronization before and after each workflow step will probably make it impossible to use this approach in systems requiring real-time execution, as in the case of botnet detection; In [Akoush et al. 2013], the authors propose a tool called HadoopProv, which was developed to capture the provenance of record-level data. It captures provenance from the intermediate keys and values created in the MapReduce process. The overhead added vary from 7% to 30%, on the map and reduce respectively. For the development of the tool it was necessary to instrument the Hadoop source code. **In summary**, these last two works use an instrumentation approach, and do not count on the functionality of WfMS for the creation and execution of workflows.

With respect to botnet detection experiments, the combination of a WfMS, provenance capture and a high performance platform would be a great contribution. Therefore, this work aims to combine the mentioned generic approaches in an innovative way, with special focus on addressing botnet experiments requirements. This proposal can address other domains, but, to the best of our knowledge, there are no initiatives reported in the botnet area. Next section describes the proposal in details.

[1] http://www.wfmc.org/

4. Architecture Overview

As in large scale scientific experiments, the botnet detection methods are characterized by being frequently executed and are constantly evolving, analyzing a large volume of data that is renewed constantly. The executions of these methods need to be carried out in a systematic way, so that it is possible to handle the highest amount of network flows as possible, minimizing the impacts of cyber-attacks. To achieve this goal, an infrastructure is required, of both hardware and software (*High Performance Infrastructure*), with high processing power, compatible with the volume of data processed, the volume of provenance data and the complexity of the activities involved in the methods.

It is necessary, as it is in large-scale scientific experiments, the development, adaptation and combination of models and tools to give support to all steps of the experiments involving botnet detection methods, i.e. creation, execution and analysis, allowing for greater collaboration between researchers, through reproduction and comparison of the experiments, and consequently increase gains in this area.

With this in mind, we created and implemented an architecture that combines and adapts scientific workflow management solutions, provenance capture and high performance data processing solutions to support such experiments. In this section, we present the architecture overview.

For better understanding of our architecture, we are considering a generic machine learning workflow to detect botnets which consists of three main steps which may involve many tasks. *Capture* is the first step, where a set of tasks may be accomplished to capture the network flow. *Curation* is the second, where the features are extracted from the flows, such as total packages, total bytes exchanged and flow duration. This step can also include tasks to clean the network flows, remove white listed domains, create indexes and store the flows in a data structure for further analysis. The last one is the *Analysis*, where one or more machine learning algorithm will be executed to identify, based on the selected flows features, suspicious machines that may belong to a botnet.

The use of a WfMS with support to provenance capture and to create and execute workflows similar to the workflow described previously would solve some of the problems presented in the previous section, such as: the experiment setup, through the workflow creation and its configuration (registry of the algorithms parameter values and input settings), the reproduction of the experiment and automation of the execution process of the method.

Moreover, another important characteristic of WfMS is that it is possible to create and maintain pre-built tasks for reuse or sharing, i.e., tasks that are ready to compose users' workflows. Therefore, in our architecture, we provide pre-built tasks that implement the most common machine leaning algorithms which are one of the most important tasks during the botnet detection process because they are directly associated to the performance in terms of time and precision of the flow classification. As we are dealing with a huge amount of data, another feature of paramount importance of WfMS that meets our needs is the possibility of the partial execution of a workflow. This will save time and resources and will be highlighted again in the following section.

But just the use of WfMS is not enough. There are issues that remain open, that WfMS alone are not able to address, such as the results of each tested workflow configura-

tion, cross-experiment analysis, methods comparison and workflow efficiency evolution. Problems that we address with others components of our architecture.

To solve the remaining problems, we designed the architecture shown in Figure 1. This architecture is divided in two stages. In the first stage, *Traffic Curation and Analysis*, the main actors are the workflows to detect botnets (WF_1, WF_2, ..., WF_N). They are responsible for tasks like select and read a source of network flows, choose the features to be evaluated by the machine learning algorithm and execute the training of the selected algorithm with their parameters. Both the values of the parameters of each task and the results found on each execution are captured and stored in the provenance database. As the training of those algorithms is a time-consuming task, we aim to store the trained algorithm as a provenance artifact for future reuse. Storing provenance of the workflows executions over time will allow the researchers to monitor their experiment, and evaluate the quality of the results of the corresponding workflow over time and to continuous refine its methods (also known as workflow evolution), either by selecting new flows feature or by changing the machine learning algorithm.

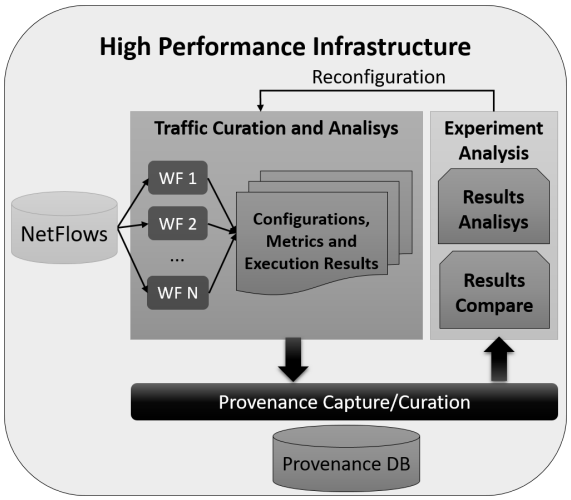


Figure 1. Architecture Overview

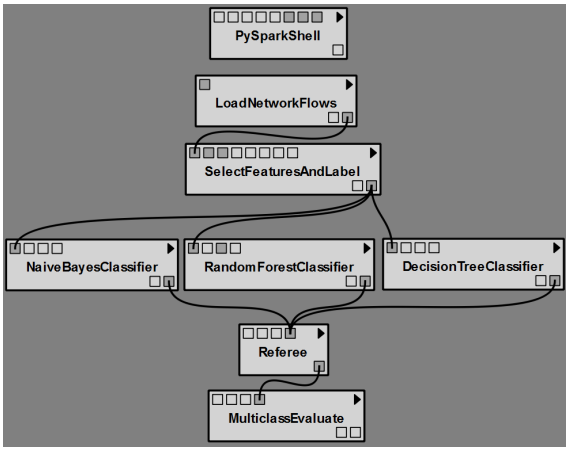


Figure 2. WfMS - VisTrails

At the second stage, *Experiment Analysis*, the researcher is able to analyze, compare and evaluate the results of different experiments. Each experiment is based on specific workflow and its set of configurations and corresponding executions. The main idea is to select the best learning algorithm to be used in a real botnet detection environment. Since all important information of the experiments are stored in the provenance database, as we will show in the next sections, we will be able to answer questions like: What is the fastest configuration of the Workflow 1(WF_1) for a given dataset? Which configuration is the most accurate? Which algorithm was used by WF_2 and with what parameter values? When the experiment was carried out and on which network flow database? A user interface is provided to facilitate the comparison of workflow executions and detection methods.

All the provenance captured, such as the workflows and all their configurations, parameter values and results, can be shared by researchers and, therefore, it is possible to repeat and compare the different methods. If sharing the dataset is not an option, it is not possible to reproduce the exact experiment, but with our architecture, the user is able to

save and share the trained algorithm as a provenance asset, so that others can execute the experiment on their own datasets and still compare the results.

The implemented architecture, which is detailed in the next section, intends to solve most of the problems presented before and contribute significantly to the development of botnet detection methods and the experiments involving them.

5. Architecture Implementation

To implement the stage *Traffic Curation and Analysis*, we started by selecting the WfMS. It should support the workflow's creation, orchestrate the execution of the experiments and capture the provenance of the workflow's tasks. We decided to choose VisTrails [Bavoil et al. 2005] because it could provide most of the functionalities of our architecture. It is a WfMS with support to graphical workflow construction and provenance capture. In VisTrails, the tasks that compose a workflow are called *modules*, which have input ports (parameters or dataset inputs) and output ports (results). Figure 2 shows the VisTrails' canvas where the workflows are created.

In addition to designing workflows and storing the provenance of their executions, VisTrails allows the partial execution of these workflows, either by a failure in the execution of a module, or by a change in a module's attribute value. For each execution, it infers which modules were successfully executed and did not change any of their parameters and mark them as *cached*. Thus, they do not need to be executed again, optimizing the workflow execution. This functionality can significantly reduce the botnet experiment execution time when paired/coupled correctly with the execution environment, as we will show later on.

Moreover, VisTrails allows users to extend its functionality by creating new modules for reuse. These new modules can then be shared with other researchers, allowing the reassembly of their workflows, in order to perform new experiments.

VisTrails already offers a set of modules that implement machine learning algorithms using the Scikit-learn library [2], but these modules are not geared to handle large volumes of data. However, one of the main requirements of our architecture is to be able to run in a high-performance environment.

One may argue that there are solutions like Weka [Witten et al. 2011] that provides an environment for tracking experiments using and enchaining machine learning activities. But Weka's purpose is not workflow management, and thus it can not address important WfMS features, like provenance capture and workflow partial execution.

Therefore, we searched for a tool that, besides being able to be implemented as a VisTrails module, could provide support for running machine learning algorithms in a high performance environment. At the same time, it should be flexible enough to allow scientists to implement their various and complex detection algorithms, avoiding any kind of limitation when they are developing their methods. With that in mind, we chose Spark [Zaharia et al. 2010], a "fast and general engine for large-scale data processing". Spark, besides being a platform for running programs in clusters, provides a library for machine learning algorithms optimized for parallel execution (ML).

[2]http://scikit-learn.org/

Spark allows users to connect to its cluster through an *interactive shell*. This enables the process of knowledge discovery to be conducted in an interactive and extremely efficient way, because while the shell is operating, the data is kept in memory, optimizing the execution of multiple operations on the same dataset.

To integrate Spark functionality with VisTrails, the development of several modules was necessary. The mapping of its functionality into modules allowed the creation of workflows inside VisTrails and hence the orchestration of their execution by VisTrails using Spark resources. These modules represent activities often undertaken in botnet detection methods based on machine learning. For every Spark feature/function we wanted to use, we implemented it as a VisTrails module. Spark will also serve as the infrastructure to run the machine learning tasks.

The main module of our implementation is the *PySparkShell*. It allows us to take advantage of one of the main feature of both tools, VisTrails *partial execution* and Spark *interactive shell*, connecting them in an innovative way. It creates an SSH channel to the Spark cluster and instantiate a PySpark shell. This shell is the channel through which commands are submitted to the Spark Cluster. Each command is created and sent by the respective module using that channel. The return of the command is also captured by the module itself when pertinent, i.e., the *MulticlassEvaluate* module captures the metrics that will be displayed later in ProvTracker. This Spark shell is kept open even after the end of the workflow execution. Thus, Spark understands that the application continues to run and maintains all data and hardware resources available for the application, allowing VisTrails to reuse them. This enables and ensures proper partial execution of the workflow optimally, executing only the necessary modules, both in case of an error in any of the modules, as in case of a parameter change/tune. To disconnect from Spark and release the resources allocated, it is necessary to close VisTrail.

Some other useful and reusable modules were also implemented to date are: *LoadNetworkFlows*; *SelectFeaturesAndLabel*; *Decision Tree*; *Random Forest*; *Naive Bayes*; *Multilayer Perception*; *Logistic Regression*; and *MulticlassEvaluate*. These modules are detailed as follows.

The *LoadNetworkFlows* module is able to read the Network Flows formatted in a CSV style. Module *SelectFeaturesAndLabel* allows the user to select each flow features he wants to be analyzed. Moreover, modules that implement wrappers to Spark provided machine learning facilities were made available, e.g., *Decision Tree*, *Random Forest*, *Naive Bayes*, *Multilayer Perception*, and *Logistic Regression*. All the Spark functions parameters are available as input ports on the corresponding modules.

Another important and required module is the *MulticlassEvaluate* module. It evaluates the results using the metrics precision, recall, f-measure, false positive rate and true positive rate for each label and the whole experiment (weighted metrics). This module also captures the results so it can be shared and analyzed.

Figure 2 shows a complete botnet detection workflow similar to the methods described at [Singh et al. 2014, Saad et al. 2011, Biglar Beigi et al. 2014, Zhao et al. 2012, Zhao et al. 2013], but using three machine learning algorithms simultaneously, *Random Forest*, *Decision Tree* and *Naive Bayes*. Note that it initiates with the *PySparkShell* module and ends with the *MultiClassEvaluate* module. In order to benefit from the Spark

facilities and to capture execution provenance, it is necessary to use these two modules. However, the user is free to compose his own workflow to address his specific needs, varying the algorithms, network load facilities, feature selection, etc.

It is worth to mention that using the Spark *interactive shell* through VisTrails (PySparkShell) it is possible to run more than one algorithm on the same dataset without the need to reload it and select its features again. As VisTrails only runs modules once per execution, and we do not close the connection to Spark, we ensure that all algorithms can access the dataset loaded in the beginning of the workflow.

All the trained models from the classifiers and *SelectFeaturesAndLabel* can also be captured and saved for later reuse or sharing, e.g. the trained algorithm can be used to classify new datasets without the need to train the algorithm every time a new dataset is available, or, as we mentioned in previous section, the trained algorithm can be shared between researchers so they can compare their results without the need to share their datasets.

At the next stage of the architecture shown in Figure 1, stage *Experiment Analysis*, subsidies are given to the scientists to analyze the experiment and results found by their methods. VisTrails allows the workflow and their provenance to be saved in a file, which can be shared with other scientists. It tracks each workflow modification, such as parametrization variations and allow the user to save it as a version into the file. For each workflow version (configuration) execution, VisTrails appends a set of provenance data to that file, such as: execution time of each module; if a module was cached; if the execution was successful; etc. To better visualize this data, we developed ProvTracker[3], a web tool that reads those files and show them in a friendly manner to the user.

Figure 3(a) shows the main page of ProvTracker. This page has a table with information about each experiment for which a *Workflow* was designed, its *Configurations*, one for each different parametrization used for its execution, and some metrics of the corresponding configuration executions: the average accuracy of the executions results; the average time spent to execute the complete workflow; the total number of executions; etc. In order to view more details about each workflow configuration, such as the parametrization values, the user has to click on the corresponding line. In the example of Figure 3(a), we can see the details of the workflow configuration *DefaultValues* from the Workflow *RandomForest*. Here we can see, for example, that the algorithm has been trained with *maxDepth* property (maximum depth of the tree) equals to 5.

ProvTracker also allows the scientist to view information about each workflow configuration execution. In the example of Figure 3 (b), ProvTracker displays information about the execution of the workflow configuration *ReducedFeatures* from the Workflow *RandomForest*. It informs: the *user* who performed the execution, the *VisTrails version* used, the *start* and *end* times of the workflow execution, if the execution was *complete*, the *total execution time* and the *accuracy metrics* of the corresponding execution results, e.g. true and false positive, precision, recall, f1, etc. It is also possible to see more details of each executed workflow module, like *execution time* and if the module was *cached* during the execution of the workflow.

It is worth to highlight that with ProvTracker, it is possible to analyze the workflow

[3]Available at http://ftoliveira.github.io/provtracker/

performance across its different configurations, and to identify the best ones. In addition, once it is possible to share the VisTrails file with the workflow configurations, other scientists can also use the tool to load all the experiment data, choose a workflow configuration and run a similar experiment on his own data. This allows scientists to compare results of different but similar experiments. Moreover, if the original input datasets are also made available, it is possible to other users to reproduce the whole experiment. Finally, the monitoring of the executions of a particular workflow over time also allows the scientist to check the effectiveness of the methodology. If its accuracy decays, he/she may review the workflow design and evolve it, in order to gain efficiency and detect botnets. It is also worth to mention that this implementation can be reproduced, since it uses open, free and mature software.

6. Results

We developed four test scenarios to explore the architecture's features and show its usability. We used our developed modules along with the CTU-13 dataset [Garcia et al. 2014a], a botnet traffic dataset captured in 2011 at the CTU University in the Czech Republic. It contains data of real botnets traffic mixed with normal traffic (reliable machines) and background traffic (untrusted machines). The CTU-13 dataset consists of thirteen captures of seven different botnet samples. Each capture was downloaded as a NetFlow file (generated with Argus) including the labels, in a CSV like style.

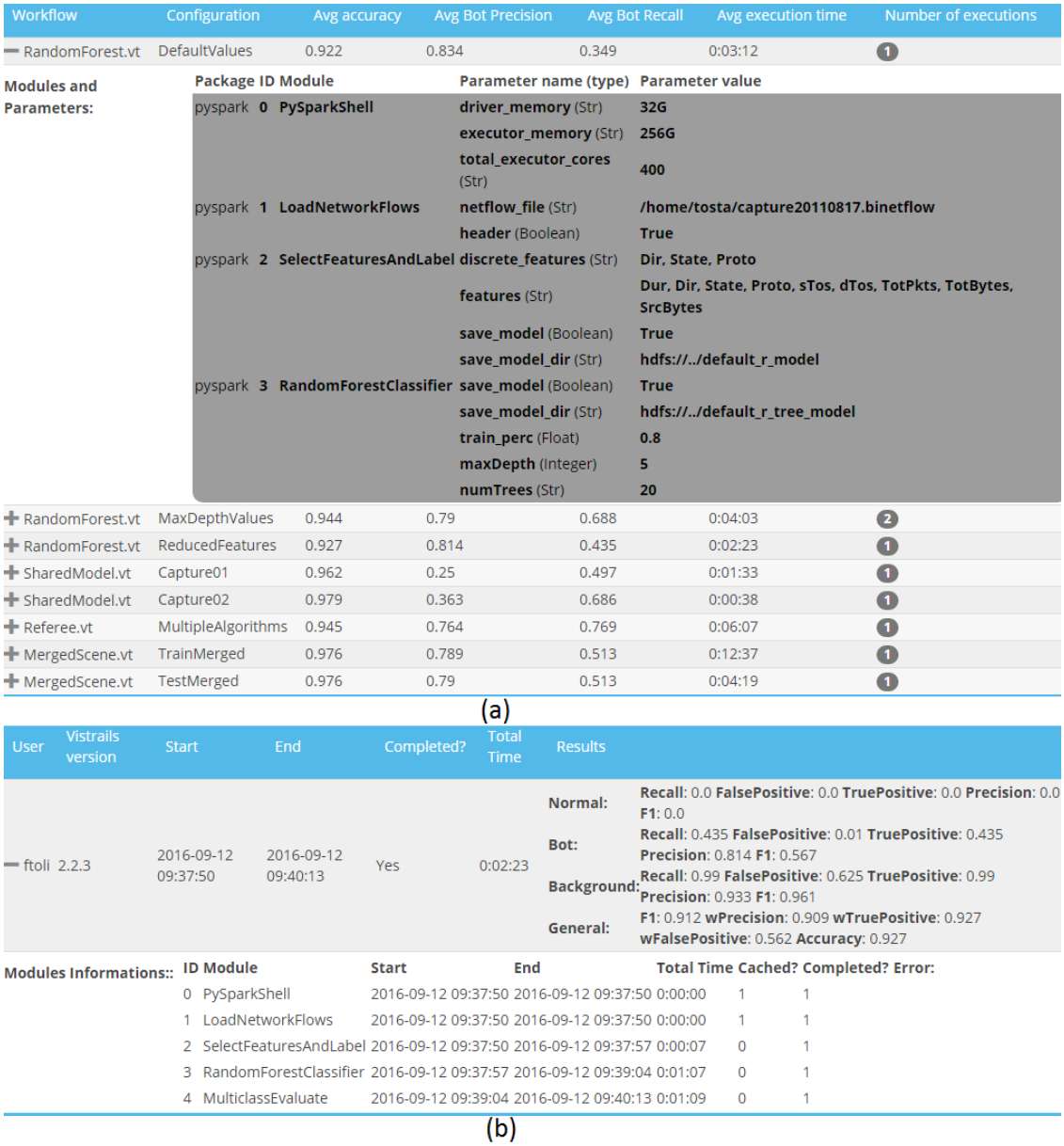
Among the information contained in each capture file, we highlight nine network flow features that we used to test our tool: Dir, State, Proto, Dur, sTos, dTos, TotPkts, TotBytes and SrcBytes. The first three of them are categorical (discrete) and the others contain continuous values. There is also the information about the category (label) of each flow, *botnet*, *normal* and *background*.

The experiment environment is composed by five machines, each one has 80 cores and 512GB of RAM. The amount of resources requested to the Spark Cluster is passed through the parameters of the *PySparkShell* module (Fig. 3(a)).

For the first scenario, we created a workflow in VisTrails called *RandomForest*, similar to the workflow shown in the Figure 2, but only with the Random Forest Classifier. The CTU-13's Capture 09 dataset (*Neris* bot) was used as input, since it has the highest number of botnet flows of all captures. We split this capture in 80% for training and 20% for testing. Internally, the module developed gets the correct % of each label.

Figure 3(a) shows that the *RandomForest* was executed with 3 different configurations. The first is called *DefaultValues*, and uses the Spark default values for the Random Forest Classifier (e.g. $numTrees = 20$, $maxDepth = 5$, etc.). For the *MaxDepthValues* and *ReducedFeatures* configurations, the maximum tree depth parameter value was changed to 30 and 5, respectively. The first two configurations used all the nine features available, while the third one used only Dur, State, Proto, TotPkts, TotBytes and SrcBytes.

Analyzing the results of the executions for the different configurations (Fig. 3(a)), the *ReducedFeatures* configuration was the fastest, while the *MaxDepthValues* was the slowest, but with the highest accuracy. Figure 3(b) shows information about the execution of the configuration *ReducedFeatures* where the modules *PySparkShell* and *LoadNetworkFlows* were cached. It happened because this configuration was executed right



Workflow	Configuration	Avg accuracy	Avg Bot Precision	Avg Bot Recall	Avg execution time	Number of executions
− RandomForest.vt	DefaultValues	0.922	0.834	0.349	0:03:12	1

Modules and Parameters:

Package	ID	Module	Parameter name (type)	Parameter value
pyspark	0	PySparkShell	driver_memory (Str)	32G
			executor_memory (Str)	256G
			total_executor_cores (Str)	400
pyspark	1	LoadNetworkFlows	netflow_file (Str)	/home/tosta/capture20110817.binetflow
			header (Boolean)	True
pyspark	2	SelectFeaturesAndLabel	discrete_features (Str)	Dir, State, Proto
			features (Str)	Dur, Dir, State, Proto, sTos, dTos, TotPkts, TotBytes, SrcBytes
			save_model (Boolean)	True
			save_model_dir (Str)	hdfs://../default_r_model
pyspark	3	RandomForestClassifier	save_model (Boolean)	True
			save_model_dir (Str)	hdfs://../default_r_tree_model
			train_perc (Float)	0.8
			maxDepth (Integer)	5
			numTrees (Str)	20

+ RandomForest.vt	MaxDepthValues	0.944	0.79	0.688	0:04:03	2
+ RandomForest.vt	ReducedFeatures	0.927	0.814	0.435	0:02:23	1
+ SharedModel.vt	Capture01	0.962	0.25	0.497	0:01:33	1
+ SharedModel.vt	Capture02	0.979	0.363	0.686	0:00:38	1
+ Referee.vt	MultipleAlgorithms	0.945	0.764	0.769	0:06:07	1
+ MergedScene.vt	TrainMerged	0.976	0.789	0.513	0:12:37	1
+ MergedScene.vt	TestMerged	0.976	0.79	0.513	0:04:19	1

(a)

User	Vistrails version	Start	End	Completed?	Total Time	Results
− ftoli	2.2.3	2016-09-12 09:37:50	2016-09-12 09:40:13	Yes	0:02:23	**Normal:** **Recall**: 0.0 **FalsePositive**: 0.0 **TruePositive**: 0.0 **Precision**: 0.0 **F1**: 0.0 **Bot:** **Recall**: 0.435 **FalsePositive**: 0.01 **TruePositive**: 0.435 **Precision**: 0.814 **F1**: 0.567 **Background:** **Recall**: 0.99 **FalsePositive**: 0.625 **TruePositive**: 0.99 **Precision**: 0.933 **F1**: 0.961 **General:** **F1**: 0.912 **wPrecision**: 0.909 **wTruePositive**: 0.927 **wFalsePositive**: 0.562 **Accuracy**: 0.927

Modules Informations::

ID	Module	Start	End	Total Time	Cached?	Completed?	Error:
0	PySparkShell	2016-09-12 09:37:50	2016-09-12 09:37:50	0:00:00	1	1	
1	LoadNetworkFlows	2016-09-12 09:37:50	2016-09-12 09:37:50	0:00:00	1	1	
2	SelectFeaturesAndLabel	2016-09-12 09:37:50	2016-09-12 09:37:57	0:00:07	0	1	
3	RandomForestClassifier	2016-09-12 09:37:57	2016-09-12 09:39:04	0:01:07	0	1	
4	MulticlassEvaluate	2016-09-12 09:39:04	2016-09-12 09:40:13	0:01:09	0	1	

(b)

Figure 3. ProvTracker interface. (a) Main page (b) Executions

after the configuration *MaxDepthValues*, and the parameters of those two modules did not change. Therefore, they were not executed again, saving processing time. We can also see more detailed metrics information, e.g. for the Bot label we have a *Precision* of 0.814.

The second scenario illustrates the sharing facility of the implemented architecture. Assuming that the input dataset could not be shared, the idea was to share the generated models. Note that the models generated by the *RandomForestClassifier* and *SelectFeaturesAndLabel* modules were saved during the execution of the *DefaultValues* configuration (save-model=true). Then, the Random Forest model was used as a classifier in the execution of the *SharedModel*, which is very similar to the *RandomForest*. This new workflow was configured to use two different CTU's captures datasets as inputs, *Capture01* and *Capture02*, respectively, and to load previously saved models.

Analyzing the execution results of the *SharedModel* (Fig. 3(a)), we can see that the average accuracy was still high and the time spent to execute it was 2-6 times faster than the original workflow (the one that generated the models). This is because the *SelectFeaturesAndLabel* and *RandomForestCassifier* modules used previously saved models, and then, since it was not necessary to create new models, they executed much faster.

For the third scenario, we tested the use of a more sophisticated workflow named *Referee*, just like the one in Figure 2, which embeds three different ML algorithms. For its configuration named *MultipleAlgorithms*, we used *maxDepth*=20 and *numTrees*= 10 for the *RandomForestClassifier* module, *maxDepth*=30 for the *DecisionTreeClassifier* module and the default values for the *NaiveBayersClassifier* module. To select the final label, we got the modal (most common) label from the three classifiers. Despite the long training time, it showed better results with respect to the precision and recall of bots.

For the fourth scenario, we wanted to test the architecture with a bigger dataset. To accomplish that, we merged all thirteen datasets into a bigger one, which has approximately 20 million flows on a 3GB file. We trained a Decision Tree Algorithm with maximum depth of 20 and named the workflow as *MergedScene*. The first configuration, named *TrainMerged*, was used to train and save the classifier model while the second one, *TestMerged*, was used to classify yet another dataset, created by concatenating this bigger dataset four times, reaching 80 millions of rows, using the saved model from the previous configuration and no cached module.

The idea was to evaluate the application of a saved model in a bigger dataset, to see if the architecture could scale and handle larger volumes of data, as in real environments, where we first train the algorithm and them use the corresponding model to classify new flows. The training time, for the *TrainMerged* configuration, as we expected, took a bit longer than the others, 12:37 minutes, 10:07 to finish the module *DecisionTreeClassifier* and 1:37 to finish the module *MulticlassEvaluate*. However, the *TestMerged* configuration execution took 4:19 minutes where 1:17 was used to evaluate the classifier and only 2:18 seconds to classify the whole dataset, almost 80 million flows!

These four scenarios showed that our architecture could address all the botnet experiments requirements: creation, execution, results analysis, experiments reproducibility and cross experiments' results comparability. In addition, it is worth to note that it was capable to scale up and handle a large amount of data.

7. Conclusion

Although there is a large number of techniques for botnet detection, they remain a threat to users and institutions, requiring the development of new detection techniques. The studies analyzed showed that the experiments performed to evaluate those techniques are still precarious, not allowing their reproduction and comparison, limiting the studies in this area. In this work, we raised the importance of a methodology to drive the execution of botnet detection experiments and the great need for frameworks and tools to support these experiments.

We specified and implemented an architecture that addresses most of the issues raised in the literature. The tests showed that it is feasible to implement and that it will benefit the experiments in this area, allowing scientists to create, execute, analyze, repro-

duce and compare their experiments, even if the input dataset is not shared. The implementation also addresses the big volume of data involved in the experiments.Moreover, the implementation contributes with a new solution for integration of VisTrails WfMS and Spark, so we can capture the provenance of the tasks involved in the experiments as well as the results achieved.

Future works include to improve the *Results Compare* module so we can facilitate the analysis of the experiment and answer questions such as: show me all workflows that used a particular input dataset. We also aim to use the architecture to capture and classify the netflows in real-time, similar to the one proposed by [Singh et al. 2014], but with provenance capture and all other benefits of our architecture. Because of the dynamism of botnets, the learning algorithms need to be trained constantly so they can evolve systemically. Upon detection of a new malicious flow, it needs to be validated by a specialist and reinserted into the dataset for retraining the algorithm. This task needs to be done systematically. As botnets flows are rare classes when compared to legitimate flows, this is feasible to be done.

References

Akoush, S., Sohan, R., and Hopper, A. (2013). HadoopProv: Towards Provenance as a First Class Citizen in MapReduce. In *Presented as part of the 5th USENIX Workshop on the Theory and Practice of Provenance*, Berkeley, CA. USENIX.

Aviv, A. J. and Haeberlen, A. (2011). Challenges in Experimenting with Botnet Detection Systems. In *Proceedings of the 4th Conference on Cyber Security Experimentation and Test*, pages 6–6, Berkeley, CA, USA. USENIX Association.

Bavoil, L., Callahan, S. P., Crossno, P. J., Freire, J., Scheidegger, C. E., Silva, C. T., and Vo, H. T. (2005). Vistrails: Enabling interactive multiple-view visualizations. In *Visualization, 2005. VIS 05. IEEE*, pages 135–142. IEEE.

Biglar Beigi, E., Hadian Jazi, H., Stakhanova, N., and Ghorbani, A. A. (2014). Towards effective feature selection in machine learning-based botnet detection approaches. pages 247–255. IEEE.

Davidson, S. B. and Freire, J. (2008). Provenance and Scientific Workflows: Challenges and Opportunities. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1345–1350, New York, NY, USA. ACM.

Freire, J., Koop, D., Santos, E., and Silva, C. (2008). Provenance for Computational Tasks: A Survey. *Computing in Science & Engineering*, 10(3):11–21.

Garcia, S., Grill, M., Stiborek, J., and Zunino, A. (2014a). An empirical comparison of botnet detection methods. *Computers & Security*, 45:100–123.

Garcia, S., Zunino, A., and Campo, M. (2014b). Survey on network-based botnet detection methods: Survey botnet detection methods. *Security and Communication Networks*, 7(5):878–903.

Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., and Myers, J. (2007). Examining the Challenges of Scientific Workflows. *IEEE Computer*, 40(12):26–34.

Hey, T., Tansley, S., and Tolle, K., editors (2009). *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Redmond, Washington, 1 edition edition.

Hofstede, R., Celeda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A., and Pras, A. (2014). Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE Communications Surveys & Tutorials*, 16(4):2037–2064.

Hogben, G., Plohmann, D., Gerhards-Padilla, E., and Leder, F. (2015). Botnets: Measurement, Detection, Disinfection and Defence - ENISA.

Korolev, V. and Joshi, A. (2014). PROB: A tool for Tracking Provenance and Reproducibility of Big Data Experiments. In *Reproduce '14. HPCA 2014*.

Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E., and den Bussche, J. V. (2011). The Open Provenance Model core specification (v1.1). *Future Generation Computer Systems*, 27(6):743–756.

Saad, S., Traore, I., Ghorbani, A., Sayed, B., Zhao, D., Wei Lu, Felix, J., and Hakimian, P. (2011). Detecting P2p botnets through network behavior analysis and machine learning. pages 174–180. IEEE.

Shi Meilin, Yang Guangxin, Xiang Yong, and Wu Shangguang (1998). Workflow management systems: a survey. volume vol.2, page 6. Publising House of Constr. Mater.

Silva, S. S., Silva, R. M., Pinto, R. C., and Salles, R. M. (2013). Botnets: A survey. *Computer Networks*, 57(2):378–403.

Singh, K., Guntuku, S. C., Thakur, A., and Hota, C. (2014). Big Data Analytics framework for Peer-to-Peer Botnet detection using Random Forests. *Information Sciences*, 278:488–497.

Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M., editors (2007). *Workflows for e-Science*. Springer London, London.

Witten, I. H., Frank, E., and Hall, M. A. (2011). *Data mining: practical machine learning tools and techniques*. Morgan Kaufmann series in data management systems. Morgan Kaufmann, Burlington, MA, 3rd ed edition. OCLC: ocn262433473.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: Cluster Computing with Working Sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, pages 10–10, Berkeley, CA, USA. USENIX Association.

Zhao, D., Traore, I., Ghorbani, A., Sayed, B., Saad, S., and Lu, W. (2012). Peer to Peer Botnet Detection Based on Flow Intervals. In Gritzalis, D., Furnell, S., and Theoharidou, M., editors, *Information Security and Privacy Research*, volume 376, pages 87–102. Springer Berlin Heidelberg, Berlin, Heidelberg.

Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A., and Garant, D. (2013). Botnet detection based on traffic behavior analysis and flow intervals. *Computers & Security*, 39, Part A(0):2 – 16.

Módulo de Proteção contra Ataques de Negação de Serviço na Camada de Aplicação: uma Análise de Qualidade de Serviço e Experiência de Usuário*

Túlio A. Pascoal[1], João H. G. Correa[1], Rafael Brayner[1], Vivek Nigam[1], Iguatemi E. Fonseca[1]

[1]Universidade Federal da Paraíba (UFPB)
Caixa Postal 5.115 - 58.051-970 – João Pessoa – PB – Brasil

{tuliopascoal,jhenrique,rafabrayner92}@gmail.com, {vivek,iguatemi}@ci.ufpb.br

Abstract. *This paper proposes a module that can defend application layer denial of service attacks. This module works in an Apache (Web) server and presents advantages when compared with other proposals in literature, as well as similar performance to a strategy that runs as a proxy. Our experimental results show that our module delivers high levels of availability, low TTS (Time To Service), memory and CPU consumption, Quality of Service and Quality of Experience even when the server is under attack.*

Resumo. *Este artigo propõe um módulo para defesa de ataques de negação de serviço na camada de aplicação. O módulo é executado em um servidor Web Apache e apresenta vantagens dentre outros existentes na literatura, com resultados similares à estratégia que opera como um proxy. Nos experimentos realizados verificou-se que o módulo proposto apresenta alta performance em termos de disponibilidade, TTS (Time To Service), consumo de memória, CPU, QoS (Quality of Service) e QoE (Quality of Experience) da aplicação em que está sendo executado.*

1. Introdução

Ataques de Negação de Serviço (DoS – *Denial of Service*) e sua versão distribuída (DDoS – *Distributed DoS*) são considerados uns dos mais perigosos e utilizados ataques contra redes e serviços. Seu objetivo é causar indisponibilidade do serviço, website ou aplicação alvo para usuários legítimos, consumindo todos os recursos disponibilizados de forma temporária ou indefinida. A grande disponibilidade e facilidade de acesso a ferramentas capazes de gerar ataques DDoS, tornam esses ataques ainda mais poderosos e comuns. Ferramentas são facilmente encontradas na Internet, como: LOIC (*Low Orbit Ion Cannon*), R.U.D.Y (*Are You Dead Yet*), *Slowloris* etc (Infosec 2013). Muitas dessas ferramentas possuem interfaces amigáveis e simples, facilitando e alavancando ainda mais o uso das mesmas, pois não requerem conhecimentos avançados de programação e redes de computadores.

Recentemente, ataques DoS na camada de aplicação (ADoS – *Application Layer DoS*), vêm sendo bastante utilizados por atacantes (Kaspersky 2016). Ataques ADoS são mais difíceis de serem detectados, pois esses ataques exploram vulnerabilidades de

*Este trabalho foi financiado e apoiado pela CAPES, CNPq e RNP.

protocolos utilizados na camada de aplicação do modelo OSI (Xie and Yu 2009), como: HTTP, HTTPS, DNS, VoIP, FTP e SMTP. Além de permitirem aos atacantes a possibilidade de focar seu ataque somente em uma aplicação ou serviço, enquanto mantém outros disponíveis, dificultando a detecção do ataque (Xie and Yu 2009). Outro fator é que o tráfego gerado por ataques do tipo ADoS *LowRate* é similar ao de usuários legítimos, dificultando sua detecção e mitigação (Dantas et al. 2014).

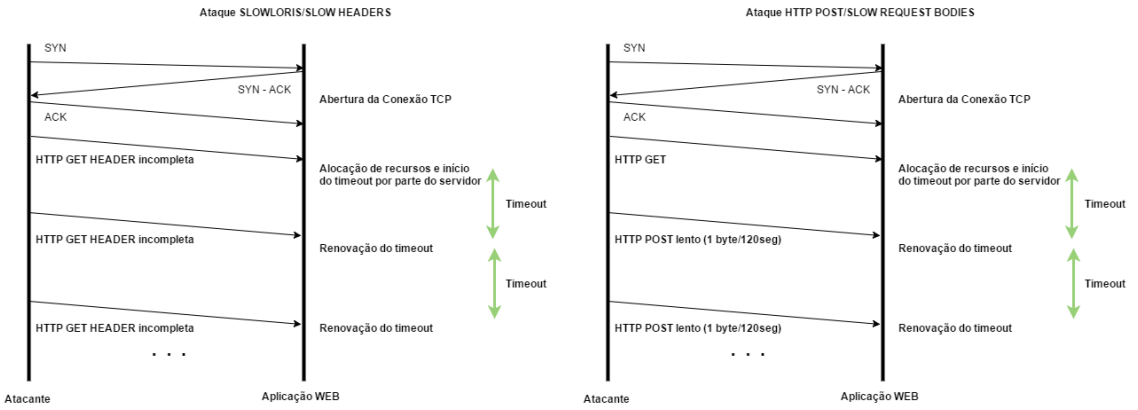
(Dantas et al. 2014) introduziram uma defesa chamada SeVen (*Selective Verification in Application Layer*) para mitigação de ataques ADoS em servidores Web. Eles demonstraram que com o SeVen ativado, o servidor Web é capaz de manter uma taxa de disponibilidade da aplicação para clientes legítimos de 95%. SeVen funciona como um *proxy*, fazendo a *interface* entre as requisições dos clientes com o servidor protegido, aumentando a complexidade da ferramenta, pois deve se preocupar, além da execução da estratégia em si, com outros aspectos como: (1) qual tipo e versão de servidor estão sendo utilizados e suas peculiaridades; (2) quais os protocolos que estão sendo utilizados, *e.g.* HTTP, HTTPS, etc; (3) segurança (integridade, confidencialidade, disponibilidade e autenticidade da ferramenta em si); (4) robustez para garantir funcionamento ininterrupto em qualquer situação, para não prejudicar o servidor que está sendo protegido; (5) limitações e dependência do sistema operacional instalado na máquina no qual a defesa está instalada. Devido a esses fatores, faz-se necessário o uso de uma abordagem menos dependente e que possa ao mesmo tempo ser eficaz e de fácil uso. Para atingir esse objetivo propomos um módulo, chamado *mod_seven* que funciona como um *mini-software* diretamente acoplado no servidor.

O Apache HTTP Server Project, mais conhecido como Apache, é o servidor Web mais popular e utilizado atualmente. Servidores Apache são usados por 55,9% entre todos os sites na rede em Dezembro de 2015 (W3tech 2016). Por ser código livre, servidores Apache fornecem a liberdade e extensibilidade de seu funcionamento, a partir da implementação de módulos (MódulosApache 2016).

Para a realização de testes que envolvem a qualidade de experiência e de serviço em websites, desenvolvemos uma ferramenta chamada webbots devido ao fato da não existência de ferramentas para a automação e execução desses tipos de testes, tanto no mercado quanto na literatura. Os webbots simulam vários usuários reais com diferentes comportamentos acessando um website, medindo parâmetros essenciais que contribuem para análise da medida de satisfação do usuário e do serviço durante a navegação.

Os objetivos e contribuições deste trabalho são, portanto: i) **mod_seven**, um módulo Apache para mitigação de ataques ADoS; ii) **webbots**, uma ferramenta para análise qualitativa da experiência de usuário e serviço; iii) **Diversos tipos de experimentos**, como *Loading Test*; *QoS*; *QoE*; replicação e comparação dos testes realizados por (Dantas et al. 2014); testes de longa duração (2 horas); e testes sob o protocolo HTTPS (não suportado pelo SeVen *proxy*); iv) **Facilitação e disseminação da utilização da estratégia**, dado que o módulo foi criado para o servidor Web mais utilizado da atualidade.

A Seção 2 apresenta a natureza, característica e funcionamento de ataques ADoS. Posteriormente, os trabalhos relacionados são apresentados na Seção 3. Na Seção 4 é apresentada a adaptação da estratégia SeVen como um módulo, bem como sua arquitetura, funcionamento e interligação com o servidor. Seção 5 demonstra a criação e carac-



(a) Simulação do ataque *Slowloris* (b) Simulação do ataque *HTTP POST*

Figura 1: Funcionamento dos ataques *Slowloris* e *HTTP POST*

terística dos *webbots* criados para análise qualitativa do módulo. A Seção 6 descreve os experimentos e discute-os. Por fim, Seção 7 expõe as conclusões e trabalhos futuros.

2. Ataques de Negação de Serviço na Camada de Aplicação

Ataques DoS consistem na interrupção temporária ou indefinida de um serviço alvo (Gu and Liu 2007), quando esses ataques exploram protocolos da camada de aplicação, podemos classificá-los como ADDoS (Xie and Yu 2009). Dentre os ataques ADDoS podemos destacar dois tipos: *Flooding* e *LowRate*. No primeiro, tem-se a geração de um enorme fluxo de tráfego, para consumir todos os recursos da aplicação, até que ela fique incapaz de atender novos clientes. O segundo, consiste na geração de tráfego similar ao de clientes legítimos, porém, utilizando-se de vulnerabilidades encontradas nos protocolos (HTTP e HTTPS, por exemplo) para manter requisições em atendimento por tempo indeterminado (Durcekova et al. 2012), assim não necessitando de grandes recursos para geração dos ataques (Dantas 2015). Existem duas grandes dificuldades para a detecção de ataques *LowRate*: sua capacidade de indisponibilizar apenas serviço(s) alvo(s), sem afetar outros disponíveis em um servidor e por seu tráfego ser bastante similar ao de clientes legítimos. Ainda mais, ferramentas que geram esses tipos de ataques são bastante simples e facilmente encontradas na Internet. Por isso, o enfoque nesses tipos de ataque neste trabalho. Abaixo tem-se a descrição e funcionamento dos dois ataques do tipo *LowRate* mais utilizados:

- **Slowloris**: Consiste no envio de requisições HTTP GET HEADER incompletas (sem os campos *CR - Carriage Return, ASCII 13, /r*, e o *LF - Line Feed, ASCII 10, /n*) no final do pacote, sempre em um certo intervalo de tempo, para forçar suas renovações, fazendo com que o servidor nunca saiba quando requisições foram finalizadas, mantendo-as no *pool* de atendimento até o valor de *timeout* pré-configurado no servidor. A partir de certo momento, todos os recursos estarão sendo consumidos pelas requisições maliciosas, indisponibilizando a aplicação. Figura 1(a) ilustra o ataque *Slowloris*;
- **HTTP POST**: Este ataque envia requisições HTTP GET HEADER completas, realizando o *TCP HANDSHAKE* com o servidor, porém envia seus dados pelo campo

BODY (via método HTTP POST) de maneira muito lenta, fazendo com que o servidor aguarde até o *timeout* configurado ou a quantidade máxima de dados no BODY de uma requisição seja atingida. Assim, essas requisições lentas tomam posse de todo o *pool* de atendimento e indisponibilizam a aplicação. Na Figura 1(b) temos o funcionamento desse tipo de ataque;

Em ambos os ataques a quantidade de tráfego gerada pelo atacante é pequena, passando assim desapercebida por defesas de monitoramento de tráfego.

3. Trabalhos Relacionados

Ataques distribuídos na camada de aplicação (ADDoS) vêm se tornando cada vez mais utilizados para indisponibilizar aplicações Web hoje em dia e seu uso não para de crescer. Além disso, (Kaspersky 2016) prevê que seu uso tende a ser alavancado. Atualmente, segundo um relatório de segurança da empresa Incapsula (Incapsula 2016), foram mitigados mais ataques na camada de aplicação do que na camada de rede. Eles também relatam que ataques ADoS estão se tornando cada vez mais pesados (470 GB de tráfego gerado em um único ataque no segundo quarto de 2016) e inteligentes.

Em (Dantas et al. 2014) uma nova estratégia para mitigação de ataques ADDoS foi implementada e criada. Nomeada de SeVen, a ferramenta é utilizada como um *proxy* e é capaz de defender servidores contra ataques DDoS. O SeVen utiliza simples funções de probabilidade e de distribuição uniforme, que quando o servidor encontra-se sobrecarregado, determinam as chances de novas requsições serem atendidas ou não. Nos experimentos realizados em (Dantas et al. 2014) e (Dantas 2015), a aplicação Web sendo protegida pela estratégia obteve disponbilidade em torno de 95% quando atacada contra dois tipos diferentes de ataques ADDoS, *Slowloris* e HTTP POST.

Outros módulos Apache existentes no mercado propõe-se a mitigar esses tipos de ataques (Monshouwer 2013) (Morimoto 2013) (Reqtimeout 2014). O *mod_antiloris* tem como estratégia a contagem de conexões simultâneas abertas por um mesmo endereço IP. Quando essa contagem superar o limiar configurado na defesa (o seu padrão é 10), o módulo rejeita todas as requisições provenientes daquele IP. O *mod_pacify_loris* (Morimoto 2013) possui a mesma estratégia de defesa (mas utiliza 50 conexões por padrão), porém ainda implementa mais duas análises: contagem de GET HEADER enviados por uma mesma requisição e a taxa de GET HEADER enviados por uma requisição por segundo, a fim de rejeitar requisições lentas, o que pode ocasionar falso positivos, prejudicando clientes legítimos que tenham conexões lentas. A abordagem desses módulos prejudicam clientes legítimos situados em redes internas, uma vez que para acesso externo todos eles utilizam um único IP público, enquanto a defesa proposta não faz distinção entre seus clientes.

Já o *mod_reqtimeout* (Reqtimeout 2014), o mais atual e utilizado dentre eles, baseia-se em uma análise mais avançada das taxas de envio dos cabeçalhos e corpo das requisições. O módulo possui uma diretiva configurável chamada *RequestReadTimeout* que possui dois parâmetros configuráveis para cada um dos campos (cabeçalho e corpo) de uma requisição, que são: *timeout* e *minrate*. O seu grande diferencial quanto aos outros módulos é que ele avalia esses parâmetros em conjunto e a cada vez que pacotes de dados de uma requisição chegam ao servidor, o módulo renova suas janelas de *timeout*. Porém, com uma simples modificação na forma de execução do ataque, é possível burlar

as estratégias do *mod_reqtimeout*, o que não ocorre no *mod_seven*

O JMeter (JMeter 2016) é uma ferramenta gratuita que permite a realização de testes de carga em websites, permitindo a avaliação da qualidade de serviço (QoS - Quality of Service) por meio de valores técnicos de rede. Porém usuários percebem o desempenho de uma forma subjetiva que é conhecida como qualidade de experiência (QoE - Quality of Experience) (Shaikh et al. 2010). Existe uma estreita relação entre a qualidade de serviço e a qualidade de experiência dos usuários (Khirman and Henriksen 2002). Devido a esta relação, o longo tempo para se obter resposta tem se refletido como um dos principais fatores para a frustração de um usuário (Schatz et al. 2013). Além desse parâmetro de avaliação, os reloads realizados na página refletem condições insatisfatórias para o usuário e pode ser considerada a única maneira imparcial de avaliação (Khirman and Henriksen 2002). Como o JMeter apenas realiza testes voltados para análise de QoS, pois não simulam usuários reais, os webbots criados neste trabalho tentam se aproximar ao máximo do comportamento de vários usuários reais diferentes bem como monitora e avalia parâmetros de qualidade para interpretar a satisfação dos usuários.

4. O Módulo Proposto

4.1. Estratégia de Defesa SeVen

O SeVen é baseado em estratégia seletiva, que seleciona quais de novas requisições, dado o momento em que o servidor encontra-se sobrecarregado, serão atendidas ou rejeitadas, a partir de funções de probabilidade. Em resumo, quando os recursos para o atendimento de novas conexões encontram-se esgotados e uma nova requisição R chega, o SeVen funciona da seguinte maneira:

1. SeVen decide a partir de uma função de probabilidade FP_1 se a nova requisição R será aceita ou não;
2. Se SeVen decide não processar R, uma mensagem erro é enviada ao usuário e a conexão com cliente é fechada (*close_connection*);
3. Se SeVen aceita processar R, ele decide a partir de uma função de probabilidade FP_2 qual conexão que atualmente está sendo processada pelo servidor deve ser substituída por R no *pool* de atendimento da aplicação.

Para melhor entender a defesa, supõe-se uma aplicação Web que possa atender, no máximo, até 4 requisições simultâneas. Iremos chamar seu *pool* de atendimento de P. Requisições são representadas pela tupla $\langle id, estado \rangle$, o valor de id identifica a requisição e $estado$ representa o estado da requisição: $PROC$ (requisição está sendo atendida) ou ESP (requisição está aguardando atendimento). Agora, supõe-se que a aplicação esteja atendendo 3 requisições simultâneas, assim temos o *pool* com a seguinte configuração:

$$\mathcal{P}_0 = [\langle id_1, \mathsf{PROC} \rangle, \langle id_2, \mathsf{PROC} \rangle, \langle id_3, \mathsf{PROC} \rangle]$$

Agora, uma nova requisição id_4, $\langle id_4, \mathsf{ESP} \rangle$ chega a aplicação. Como o *pool* de atendimento ainda possui espaço para atender novas requisições, id_4 é simplesmente adicionada e atendida:

$$\mathcal{P}_1 = [\langle id_1, \mathsf{PROC} \rangle, \langle id_2, \mathsf{PROC} \rangle, \langle id_3, \mathsf{PROC} \rangle, \langle id_4, \mathsf{PROC} \rangle]$$

Logo após, uma nova requisição id_5, $\langle id_5, \mathsf{ESP} \rangle$ é recebida pela aplicação, o SeVen detecta que o *pool* está na sua capacidade máxima e deve decidir se id_5 será atendida ou não, isso é feito pelo resultado obtido na função de probabilidade FP_1 (seus detalhes não

entram no contexto desse artigo). Caso FP_1 decida que a requisição id_5 não será atendida, uma mensagem de erro é enviada e a conexão fechada. Assim:

$$\mathcal{P}_2 = [\langle id_1, \mathsf{PROC}\rangle, \langle id_2, \mathsf{PROC}\rangle, \langle id_3, \mathsf{PROC}\rangle, \langle id_4, \mathsf{PROC}\rangle]$$

Porém, caso FP_1 decida que a nova requisição id_5 deva ser processada, uma segunda função, de distribuição uniforme (FP_2) decidirá qual requisição dentre as que estão sendo processadas atualmente deve ser eliminada do *pool* de atendimento. Supondo que a FP_2 decidiu que a requisição id_3 deva ser eliminada, temos:

$$\mathcal{P}_3 = [\langle id_1, \mathsf{PROC}\rangle, \langle id_2, \mathsf{PROC}\rangle, \langle id_5, \mathsf{PROC}\rangle, \langle id_4, \mathsf{PROC}\rangle]$$

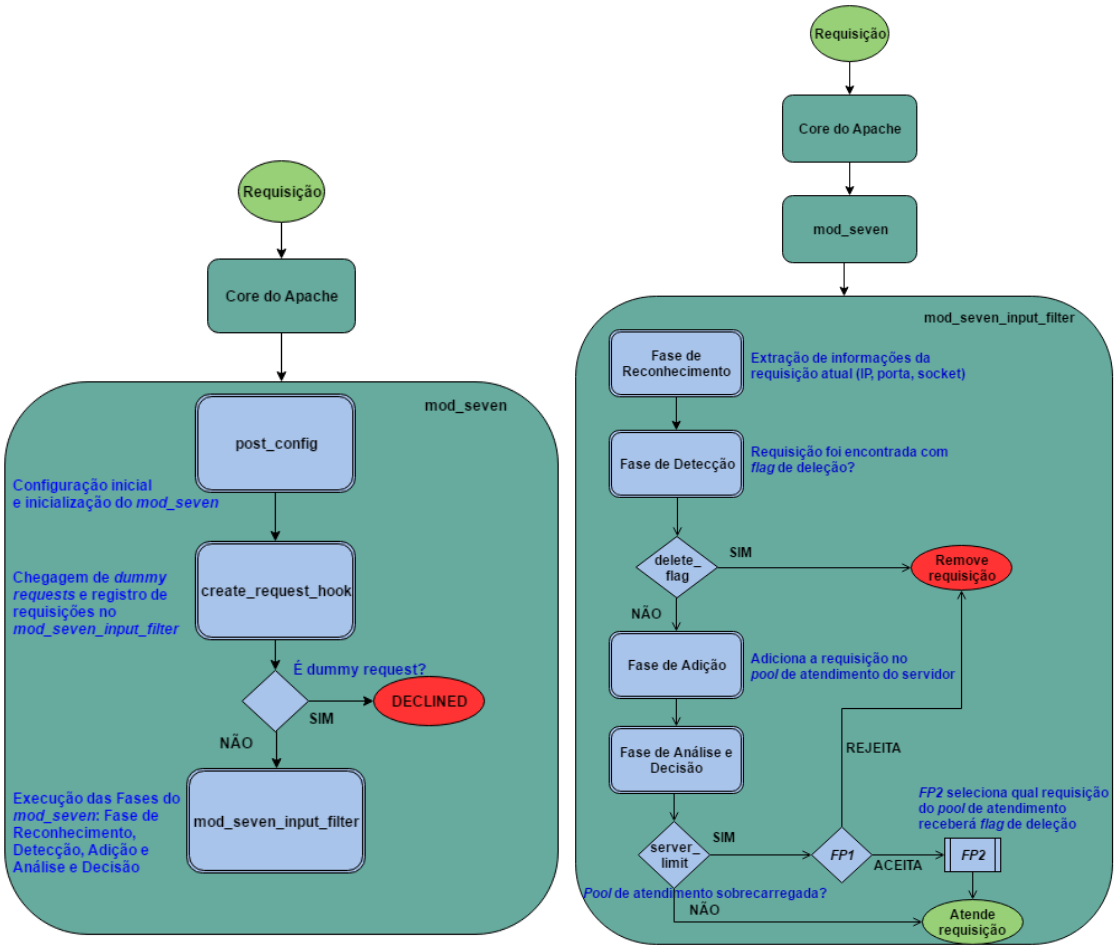
SeVen funciona pois quando um servidor encontra-se sobrecarregado, ele muito provavelmente estará sofrendo um ataque DoS. Consequentemente, o seu *pool* de atendimento estará repleto de conexões atacantes, ou com mais atacantes do que clientes. Portanto, quando o SeVen decide processar uma nova requisição, a chance de remover uma conexão atacante do *pool* é muito maior do que de um cliente legítimo (Dantas et al. 2014).

4.2. O Módulo Apache Proposto: *mod_seven*

De acordo com (Kew 2007), o Apache HTTP Server é composto por um pequeno *core* e um conjunto de módulos, que acrescentam funcionalidades ao Apache. O processo de atendimento de requisições em servidores Apache funciona da seguinte forma: assim que uma nova requisição chega ao servidor, o *core* do Apache recebe-a e realiza uma checagem de quais módulos ativados no servidor tem interesse de processar a requisição e a repassa. Dessa forma, os módulos executam suas funções extras em conjunto com o atendimento normal do servidor *web*.

A primeira ação do módulo *mod_seven* é auto declarar-se ao *core* do Apache, para deixá-lo informado sobre sua existência, ativação e registro de seus ganchos. Isso é realizado pela diretiva *AP_MODULE_DECLARE_DATA mod_seven*. Ganchos funcionam como métodos que informam ao *core* do Apache quais tipos de requisições e em qual etapa (processamento da requisição ou geração de conteúdo, por exemplo) o módulo deseja operar. O *mod_seven* possui três ganchos: (1) o *ap_hook_post_config*, utilizado para recolher informações do servidor, como tamanho do *pool* de atendimento, criação e alocação de *threads* da defesa; (2) o *ap_hook_create_request* para detecção de *dummy requests* (Hayden 2008), um *bug* do Apache encontrado durante a implementação e registro das requisições no filtro do módulo; (3) o *mod_seven_input_filter* que é o filtro que executa praticamente toda a estratégia. Quando uma requisição é registrada em um filtro, todo e qualquer dado enviado pela requisição será analisado e processado pelo filtro. Para simplificação e melhor entendimento, o funcionamento do *mod_seven_input_filter* do módulo foi dividido em 4 fases distintas, explicadas abaixo:

- **Fase de Reconhecimento**: Nessa fase o filtro extrai informações da requisição (endereço IP, porta e *socket* da conexão), e aloca em uma variável-estrutura interna da APR do tipo *worker_score* para representar a requisição no *pool* de atendimento do Apache, chamado de *scoreboard*;
- **Fase de Detecção**: Aqui acontece uma adaptação necessária à estratégia para adequar-se ao comportamento de Módulos Apache. Como o Apache não permite a extração e manuseamento direto de requisições que se encontram no *scoreboard*, essa fase realiza uma varredura no *scoreboard* atual da aplicação verificando



(a) Organização e ordem de fluxo de execução do *mod_seven*

(b) Organização e ordem de fluxo das fases do *mod_seven_input_filter*

Figura 2: Divisão de processos e execução do *mod_seven*

se a requisição atual (que está sendo processada pelo filtro) está com *flag* que foi selecionada para ser removida pela estratégia, caso positivo, a conexão daquela requisição será fechada imediatamente; caso negativo, o fluxo do módulo continua para a próxima fase, a Fase de Adição;

- **Fase de Adição**: É uma fase simples e direta, após colher as informações da requisição e verificar que a mesma não encontra-se selecionada para deleção, o módulo conclui que ela está apta a ser atendida e processada, e a requisição é adicionada ao *pool* de atendimento do Apache.
- **Fase de Análise e Decisão**: É a fase mais completa e que aplica toda a lógica da estratégia. Uma contagem de requisições no *pool* de atendimento é realizada, similar ao da Fase de Detecção, para concluir se o servidor encontra-se sobrecarregado ou não, utilizando uma variável para comparar com o valor do parâmetro *server_limit* (que representa o máximo de conexões simultâneas que o servidor pode atender). Se o servidor estiver sobrecarregado, a função de probabilidade FP_1 decide a aceitação ou não da requisição. Caso a requisição seja rejeitada pela estratégia, sua conexão é automaticamente fechada usando *APR_DECLINED*, *ap_conn_close_close*

e *apr_socket_close*. Caso contrário, a requisição atual será atendida, para isso a função de distribuição uniforme FP_2 (origem da característica seletiva da estratégia) seleciona uma requisição para ser substituída do *pool* de atendimento. Após a escolha, o módulo resgata informação do *worker_score* escolhido, e adiciona a *flag* de deleção, assim, quando qualquer dado referente àquela requisição chegar ao servidor, a mesma será rejeitada automaticamente na Fase de Detecção.

Enquanto a aplicação não se encontra sobrecarregada, o módulo simplesmente executa as Fases de Reconhecimento e Adição. Na Figura 2(a) tem-se o fluxo de funcionamento e processos do *mod_seven* realizados pelos seus ganchos e métodos internos e na Figura 2(b) tem-se uma visão geral da divisão e execução de processos ocorridos nas distintas fases do *mod_seven_input_filter*.

5. Webbots - Medindo Qualidade de Experiência e de Serviço

Qualidade de Serviço (do inglês Quality of Service - QoS) na web está diretamente relacionada ao desempenho perceptível pelos usuários. Essa qualidade pode ser medida quantitativamente por meio da análise de diferentes aspectos do serviço como taxa de erro, taxa de bits, atraso de transmissão, disponibilidade, taxa de transferência, dentre outros. QoS são tipicamente parâmetros de rede. No entanto, usuários percebem o desempenho de forma subjetiva, eles não estão interessados em valores técnicos da rede, e sim em obter resposta dentro de um tempo razoável. Essa percepção subjetiva é conhecida como Qualidade de Experiência (do inglês Quality of Experience - QoE) (Shaikh et al. 2010).

O JMeter (JMeter 2016) é uma das mais utilizadas ferramentas para verificação de desempenho de websites, por apresentar uma vasta gama de configurações, além de ser uma ferramenta já conceituada e gratuita. JMeter é um projeto do Apache Software Foundation que permite realizar testes simulando vários usuários realizando tarefas em website. Ele permite especificar diversos tipos de parâmetros, dentre eles, o número de usuários a serem criados, o tempo total em que esses usuários devem ser criados e o número de vezes que cada usuário vai realizar as tarefas. Além disso, ele também permite definir o fluxo das atividades, como por exemplo, executar o login apenas uma vez. No entanto, o JMeter não é capaz de realizar testes mais elaborados, principalmente em relação a qualidade de experiência e serviço da aplicação, quando tratando-se na simulação comportamental de usuários reais.

Devido ao fato de que a qualidade de serviço está intimamente ligada a qualidade de experiência dos usuários (Khirman and Henriksen 2002), desenvolveu-se os webbots para simular um grande número de usuários reais, realizando tarefas em um website, e medir a satisfação dos mesmos durante o processo de navegação. Os parâmetros de qualidade escolhidos para mensurar a satisfação foram o valor máximo de tempo que um robô espera por uma resposta do servidor e o número máximo de *"reloads"* realizados durante a navegação. O primeiro está relacionado com um dos maiores problemas que tem sido enfrentado pelos usuários na web e que causa mais frustração, que é o longo tempo para se obter resposta (Schatz et al. 2013) (Selvidge 2003). O segundo está relacionado com as condições insatisfatórias no website que podem fazer com que o usuário pressione o botão de recarregar. Apesar dessa não ser a única maneira de avaliar a insatisfação do usuário em relação qualidade da comunicação, pode ser considerada a única maneira de se obter uma forma de medida independente e imparcial (Khirman and Henriksen 2002).

Os webbots oferecem alguns diferenciais dos quais podem-se listar três principais características que são exclusivas: Primeiro, os webbots tentam se aproximar o máximo possível de usuários reais, interagindo com as páginas, de maneira que as requisições são realizadas em intervalos aleatório de tempo a fim de simular diferentes tipos de usuários, desde o mais lento ao mais rápido, ao contrário do JMeter que não oferece essa funcionalidade. Segundo, quando os webbots estão realizando uma sequência de tarefas e se deparam com um erro causado pela indisponibilidade do , eles tentam realizar a mesma tarefa novamente por um número 'x' de vezes. Esse número é considerado como o máximo aceitável de tentativas por um usuário. Por fim, eles geram um relatório estatístico baseado nas suas ações de sucesso ou de falha (que estão relacionados aos parâmetros de qualidade de serviço da perspectiva do usuário). Neste relatório cada robô informa os tempos de resposta para cada ação, o número de tentativas para cada uma delas, o tempo total para realizar as tarefas, bem como o status de sucesso ou falha na realização das tarefas.

Enfim, os webbots são uma ferramenta inédita capaz de realizar testes e análises qualitativas de um website a partir da simulação de usuários "robôs"que simulam usuários reais interagindo com a aplicação e colhendo métricas importantes para o mensarumento da qualidade de navegação.

6. Experimentos e Resultados

6.1. Cenário e Configuração dos Experimentos

Os testes foram realizados usando três máquinas em dois diferentes *Campus* da Universidade Federal da Paraíba (UFPB). Duas máquinas distintas situadas no *Campus* V gerando o tráfego de clientes legítimos e atacantes, e uma máquina no *Campus* I hospedando a página Web padrão do Apache. As máquinas para geração de tráfego possuem processador Intel i5-3470 de 3.20Ghz e 4GB de RAM e o servidor um Intel Xeon E5-2620 de 2.00GHz e 8GB de RAM. O objetivo dessa configuração de cenário é de simular, com um maior grau de realidade, um ataque DoS, em que o tráfego situa-se em redes distintas e separadas fisicamente. Para geração do tráfego cliente utilizamos a ferramenta Siege (ferramenta de benchmark para medir desempenho de aplicações Web (Siege 2015)) e para o tráfego atacante duas ferramentas: *Slowloris* (Slowloris 2013) e *Slowhttptest* (Slowhttptest 2013).

O servidor foi configurado com um *timeout* (tempo, em segundos, que o servidor aguarda para receber dados de requisições em uma mesma conexão) de 40 segundos e *MaxRequesWorkers* (número máximo de requisições simultâneas atendidas pelo servidor) de 200, essa configuração representa um servidor de pequeno/médio porte. À quesito de comparação de resultados, a configuração e cenários dos experimentos realizados nesse trabalho foram replicados de acordo com os realizados por (Dantas et al. 2014).

- **Tráfego atacante**: 250 atacantes para cada tipo de ataque, enviando requisições a cada 35 segundos. Aproximadamente 7,14 requisições por segundo;
- **Tráfego de clientes legítimos**: 100 clientes simultâneos enviando requisições em um intervalo de 0 a 3 segundos, a fim de melhor simular um tráfego Web legítimo. Aproximadamente 10 requisições por segundo.
- **Duração**: três repetições para os testes de 5 minutos e uma para os testes de 2 horas;

Tabela 1: Comparação dos resultados entre *mod_seven* e SeVen *proxy*

	mod_seven		**SeVen *proxy***	
	Disponibilidade	TTS	Disponibilidade	TTS
Sem Ataque	100%	0,03s	100%	0,03s
Slowloris	98,7%	0,07s	97,3%	0,05s
HTTP POST	95,1%	0,02s	94,5%	0,06s

Nós utilizamos os seguintes parâmetros como métricas de desempenho: i) **Disponibilidade**: Porcentagem dos clientes atendidos com sucesso; ii) **TTS**: Tempo médio de resposta para cada requisição; iii) **Consumo de memória e CPU**: Porcentagem do consumo médio de memória e CPU durante o teste;

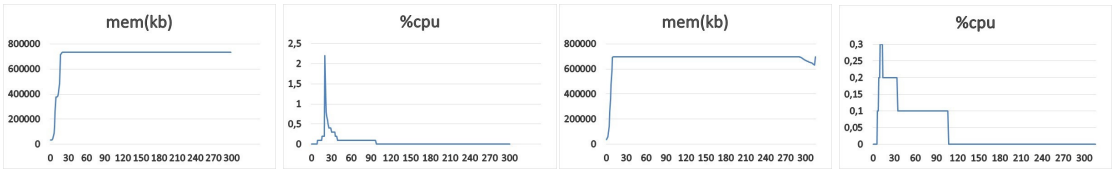
Um segundo tipo de experimentos foi realizado utilizando os webbots discutidos na seção 5 para uma melhor análise do módulo em quesito de *QoS* e *QoE* da aplicação Web. Para alcançar esse objetivo, uma aplicação Web que simula a página do SISU (Sistema de Seleção Unificada (veja `http://sisu.mec.gov.br/`) para inscrições de alunos em cursos superiores das universidades brasileiras, que ocorrem semestralmente e recebem um enorme tráfego em um curto período de tempo; e que já enfrentaram casos de ataques DoS (RNP 2016). O *site* consiste de cinco páginas com formulários que devem ser preenchidas pelos alunos, desde a primeira página de *Login*, Escolha de Cursos (duas opções), Modalidade até a quinta e última página, que finaliza a inscrição. Em algumas páginas acontecem consultas ao banco de dados da aplicação, para listar universidades e cursos, por exemplo. Para hospedar a página foi utilizado o Apache Tomcat, versão 8.0.14 devido a necessidade de executar procedimentos Java, não suportado pelo Apache Server. Também foi necessário a instalação de um servidor Apache, funcionando como um *proxy* (com o *mod_seven* instalado), recepcionando requisições e repassando-as ao Tomcat, dado que o módulo não funcionaria diretamente no Tomcat.

Cada webbot preenche cada um dos cinco formulários necessários para finalizar a inscrição em uma velocidade randômica e compatível com a humana, ou seja, alguns robôs podem ser mais rápidos que outros. Quando encontram algum tipo de erro ou a resposta recebida ultrapassou **20 segundos**, o webbot faz um *"reloads"* na página para reiniciar a inscrição a partir do último formulário preenchido com sucesso. Robôs realizam no máximo até **três** *"reload"* em uma mesma página ou **10** no total (entre as cinco páginas) antes de desistirem de realizar a inscrição. Se um robô não desistiu da inscrição pelos fatores acima explicados, eles concluíram a inscrição com sucesso.

Tabela 2: Resumo dos resultados nos experimentos de 2 horas do *mod_seven*

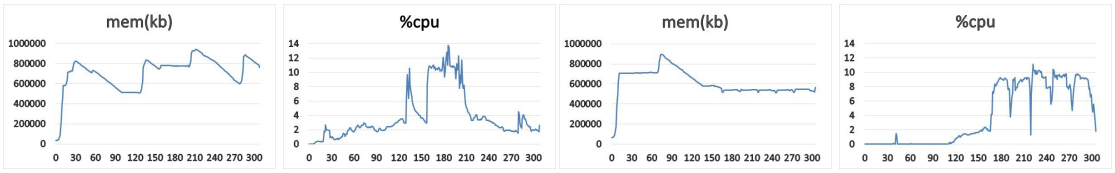
	Protocolo HTTP		**Protocolo HTTPS**	
	Disponibilidade	TTS	Disponibilidade	TTS
Slowloris	97,5%	0,14s	91,6%	0,09s
HTTP POST	91,2%	0,05s	92,4%	0,07s

Por SeVen se tratar de uma defesa agnóstica, os testes com os robôs medem o desempenho da defesa nos casos em que (mesmo com menores chances de ocorrerem) clientes legítimos tenham suas conexões fechadas durante a realização da inscrição. Assim, simulando um cenário real de aplicações Web, principalmente aquelas que precisam ar-



(a) Memória e CPU no ataque *Slowloris* (b) Memória e CPU no ataque *HTTP POST*

Figura 3: Gráfico do consumo de memória e CPU com ataque e sem *mod_seven*



(a) Memória e CPU no ataque *Slowloris* (b) Memória no ataque *HTTP POST*

Figura 4: Gráfico do consumo de memória e CPU com ataque e com *mod_seven*

mazenar sessões dos usuários e com vários formulários para preenchimento, por exemplo: *Booking* de passagens aéreas, operações em *Internet Banking*, *E-Commerce* etc. Nesses experimentos, as configurações dos servidores e o tráfego atacante foram as mesmas dos testes anteriores. O tráfego dos webbots foi de 1.000 robôs por experimento, variando sua taxa de criação em 10, 50 e 100 por minuto por teste realizado.

6.2. Resultados e Discussão

Comparando com o SeVen *proxy*, os resultados (Tabela 1) do módulo mostram um pequeno aumento da disponibilidade da aplicação (acréscimo médio de 1%) quando comparado com os resultados nos mesmos experimentos realizados em (Dantas et al. 2014). Além de possibilitar a aplicação da estratégia no protocolo HTTPS (não suportado pelo SeVen *proxy*) e em qualquer outro, desde que suportado pelo Apache. Outra vantagem é que a estratégia pode ser melhor difundida dado a preferência de uso dos servidores Apache pela comunidade, além de fácil instalação e utilização.

Tabela 3: Disponibilidade, TTS, consumo de memória e CPU dos testes realizados

	Sem mod_seven				**Com mod_seven**			
	Disponibilidade	TTS	Memória	CPU	Disponibilidade	TTS	Memória	CPU
Sem Ataque	100%	0,01s	0,9%	7,8%	100%	0,03s	0,9%	8,7%
Slowloris	0,0%	∞	17,5%	0,0%	98,7%	0,07s	18,2%	2,6%
HTTP POST	0,0%	∞	16,6%	0,0%	96,6%	0,03s	13,5%	1,8%

Pela Tabela 3 percebe-se que o *mod_seven* não influencia na disponibilidade da aplicação em situações normais (sem ataque), identifica-se um pequeno aumento do consumo de CPU (de 7,8% para 8,7%) e de 0,02 segundos no TTS, devido aos processamentos adicionais realizados pelo módulo. Analisando os cenários com ataque, verifica-se a eficiência do *mod_seven*, que manteve a aplicação com uma disponibilidade de 98,7% e 96,6% nos ataques *Slowloris* e *HTTP POST*, respectivamente, e com baixo TTS. Pela Figura 4 nota-se que, o consumo de memória quando utilizando o módulo não é elevado em ambos os ataques. Outro fator interessante é o consumo de CPU ser nulo (com pico de 0,3%) quando sob ataque e sem módulo, isso é uma prova da eficiência do ataque, pois, uma vez que a aplicação está indisponível, a mesma não processa mais nenhuma requisição, fazendo com que não haja mais consumo de CPU (ver (Figura 3). Já no

cenário com *mod_seven* (Figura 4) temos o consumo de recursos intercalados, mostrando que o servidor encontra-se ativo (atendendo requisições).

Tabela 4: Resumo dos resultados dos experimentos com os robôs

Taxa de Criação: 10 robôs/minuto				
Ataque	Apache + Tomcat		Apache + Tomcat + *mod_seven*	
	Sucesso	Falharam	Sucesso	Falha
Sem Ataque	1000	0	1000	0
Slowloris	0	1000 (2/998)	947	53 (23/30)
HTTP POST	0	1000 (122/878)	999	1(0/1)

Taxa de Criação: 50 robôs/minuto				
Ataque	Apache + Tomcat		Apache + Tomcat + *mod_seven*	
	Sucesso	Falharam	Sucesso	Falha
Sem Ataque	1000	0	1000	0
Slowloris	0	1000 (0/1000)	929	71 (12/59)
HTTP POST	0	1000 (58/942)	943	57 (4/53)

Taxa de Criação: 100 robôs/minuto				
Ataque	Apache + Tomcat		Apache + Tomcat + *mod_seven*	
	Sucesso	Falharam	Sucesso	Falha
Sem Ataque	1000	0	1000	0
Slowloris	0	1000 (0/1000)	913	87 (18/69)
HTTP POST	0	1000 (1/999)	937	63 (1/62)

Nos testes de 2 horas (HTTP e HTTPS), confirmou-se o desempenho do *mod_seven*. Verifica-se uma pequena queda na disponibilidade para o ataque *HTTP POST* (Tabela 2), uma vez que a taxa de requisições por segundo aumentou (devido ao uso da ferramenta *Slowhttptest*), transformando o ataque em um *mini-flooding*, dando indícios que o *mod_seven* possa obter bons resultados contra ataques do tipo *Flooding*. No HTTPS, verificou-se que o *Slowloris* mostrou-se incapaz de realizar o ataque, por isso foi utilizado o *Slowhttptest*. Houve uma redução da disponibilidade e um pequeno aumento no TTS, sobretudo porque o protocolo HTTPS aplica mais métodos de segurança e criptografia (contudo, uma maior e mais específica análise deve ser aplicada em relação ao *overhead* causado pelo HTTPS). Outra importante conclusão é da robustez e bom gerenciamento de *threads* e processos do módulo, uma vez que suportou uma carga elevada (recebeu cerca de 79.945 pacotes atacantes e 422.511 requisições de clientes nos testes de 2 horas).

Em relação ao *mod_antiloris* e *mod_pacify_loris*, o *mod_seven* se mostra uma defesa mais inteligente, pois não discrimina as requisições recebidas, todas possuem as mesmas chances de serem atendidas. Esses módulos utilizam uma estratégia discriminatória, dado que bloqueiam requisições baseado no IP público da conexão. Assim, prejudicando

usuários que estejam numa mesma rede interna (Abordagem bastante utilizada em redes de grandes organizações, no qual um único IP pode representar mais de uma máquina da sua rede interna). Pela tática da taxa de cabeçalhos por segundo do *mod_pacify_loris*, ele pode erroneamente rejeitar requisições de clientes legítimos, porém com conexões lentas.

Já o *mod_reqtimeout*, apesar de possuir uma estratégia mais elaborada, baseada em *timeouts* e *minrate* renováveis, possui vulnerabilidades. Atacantes podem utilizar estratégias para detectar esses valores a partir de experimentos prévios ao ataque. Por exemplo, mandando requisições em diferentes intervalos de tempo, e verificando o comportamento e as respostas do servidor, encontra-se um valor aproximado do tempo que suas conexões mantêm-se em atendimento até que comecem a ser rejeitadas. Esse valor de tempo é provavelmente o *timeout* configurado na defesa. Com essa informação, realiza-se o ataque com o *timeout* de suas conexões com um valor próximo ao detectado, assim, renovando suas conexões antes de serem removidas, mantendo-as indefinidamente em atendimento, burlando a defesa do *mod_reqtimeout*.

Tabela 4 expõe os resultados obtidos nos testes com os robôs no SISU, no qual podemos ver que quando a aplicação não estava protegida pelo *mod_seven*, a maioria dos robôs não conseguiram realizar a inscrição com sucesso, na sua grande maioria, devido ao longo tempo de espera para receeber a resposta do servidor. Para os robôs que falharam foi utilizado a seguinte notação: $T(N_R,T_R)$, T é o total de robôs que falharam, N_R, a quantidade de robôs que falharam pelo número máximo de *"reload"* e T_R pelo tempo de espera de resposta do servidor. Com o *mod_seven*, todavia, a grande maioria dos robôs conseguiram realizar suas inscrições em todos os casos realizados. Mesmo no cenário de maior tráfego (100 robôs/minuto) a taxa de sucesso foi superior a 91%.

7. Conclusão e Trabalhos Futuros

Este artigo apresenta uma solução para uns dos maiores problemas atuais na Internet, ataques DoS. Mas que também facilita o uso e gerenciamento (a partir da abordagem de módulos) de uma defesa eficaz, chamada SeVen. Além de proporcionar seu uso em diversos tipos de protocolos (SeVen *proxy* (Dantas 2015) só funciona no HTTP), desde que suportados pelo Apache (servidor Web mais utilizado atualmente). O *mod_seven* obteve melhores resultados em diversos cenários de experimentos e quando comparado com o SeVen *proxy* e outros módulos de defesa. Concomitantemente, com baixo consumo de CPU e memória e garantindo a *QoS* da aplicação, verificado a partir de experimentos com robôs simulando inscrições no *site* do SISU, um cenário mais complexo, com páginas contendo vários formulários, consulta em banco de dados e que necessitam de conexões persistentes dos usuários. Como trabalho futuro, objetiva-se testar o *mod_seven* na mitigação de ataques DoS em outros protocolos suportados pelo Apache; testá-lo contra ataques *Slowread*, ataque do tipo *Lowrate* mais recente (Park et al. 2015), bem como ataques *Flooding*. Ainda, estudar o comportamento do módulo e dos ataques em outros tipos de MPMs do Apache.

Referências

[Dantas 2015] Dantas, Y. G. (2015). Estratégias para tratamento de ataques de negação de serviço na camada de aplicação em redes ip. Master Thesis in Portuguese.

[Dantas et al. 2014] Dantas, Y. G., Nigam, V., and Fonseca, I. E. (2014). A selective defense for application layer ddos attacks. In *Intelligence and Security Informatics Conference (JISIC), 2014 IEEE Joint*, pages 75–82. IEEE.

[Durcekova et al. 2012] Durcekova, V., Schwartz, L., and Shahmehri, N. (2012). Sophisticated denial of service attacks aimed at application layer. In *ELEKTRO, 2012*, pages 55–60. IEEE.

[Gu and Liu 2007] Gu, Q. and Liu, P. (2007). Denial of service attacks. *Handbook of Computer Networks: Distributed Networks, Network Planning, Control, Management, and New Trends and Applications*, 3:454–468.

[Hayden 2008] Hayden, M. (2008). *Apache 2.2: internal dummy connection*. `https://major.io/2008/09/23/apache-22-internal-dummy-connection/`. Acessado em: 25 de Julho de 2015.

[Incapsula 2016] Incapsula (2016). *DDoS Threat Landscape Report 2015-2016*. `https://lp.incapsula.com/DDoSThreatLandscapeReport2015-2016_LP.html`. Acessado em: 27 de Setembro de 2016.

[Infosec 2013] Infosec (2013). *Layer 7 DDoS Attacks*. `http://resources.infosecinstitute.com/layer-seven-ddos-attacks/`. Acessado em: 25 de Outubro de 2015.

[JMeter 2016] JMeter (2016). *The Apache Software Foundation - Apache JMeter*. `http://jmeter.apache.org/`. Acessado em: 01 de Dezembro de 2016.

[Kaspersky 2016] Kaspersky (2016). *Kaspersky DDoS Intelligence Report for Q1 2016*. `https://securelist.com/analysis/quarterly-malware-reports/74550/kaspersky-ddos-intelligence-report-for-q1-2016/`. Acessado em: 22 de Agosto de 2016.

[Kew 2007] Kew, N. (2007). *The Apache modules book: application development with Apache*. Prentice Hall Professional.

[Khirman and Henriksen 2002] Khirman, S. and Henriksen, P. (2002). Relationship between quality-of-service and quality-of-experience for public internet service. In *In Proc. of the 3rd Workshop on Passive and Active Measurement*.

[Monshouwer 2013] Monshouwer, K. (2013). *mod_antiloris*. `https://sourceforge.net/projects/mod-antiloris/`. Acessado em: 10 de Agosto de 2015.

[Morimoto 2013] Morimoto, S. (2013). *mod_pacify_loris*. `http://mod-pacify-slowloris.googlecode.com/svn/trunk/mod_pacify_loris.c`. Acessado em: 10 de Agosto de 2015.

[MódulosApache 2016] MódulosApache (2016). *Developing modules for the Apache HTTP Server 2.4*. `http://httpd.apache.org/docs/2.4/developer/modguide.html`. Acessado em: 07 de Outubro de 2016.

[Park et al. 2015] Park, J., Iwai, K., Tanaka, H., and Kurokawa, T. (2015). Analysis of slow read dos attack and countermeasures on web servers. *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, 4(2):339–353.

[Reqtimeout 2014] Reqtimeout (2014). *mod_reqtimeout*. `https://httpd.apache.org/docs/2.4/mod/mod_reqtimeout.html`. Acessado em: 11 de Agosto de 2015.

[RNP 2016] RNP (2016). *RNP participa da operação de monitoramento do Sisu*. `https://www.rnp.br/noticias/rnp-participa-operacao-monitoramento-sisu`. Acessado em: 01 de Junho de 2016.

[Schatz et al. 2013] Schatz, R., Hoßfeld, T., Janowski, L., and Egger, S. (2013). From packets to people: quality of experience as a new measurement challenge. In *Data traffic monitoring and analysis*, pages 219–263. Springer.

[Selvidge 2003] Selvidge, P. (2003). Examining tolerance for online delays. *Usability News*, 5(1):1–5.

[Shaikh et al. 2010] Shaikh, J., Fiedler, M., and Collange, D. (2010). Quality of experience from user and network perspectives. *annals of telecommunications-annales des telecommunications*, 65(1-2):47–57.

[Siege 2015] Siege (2015). *Linux man page: siege - An HTTP/HTTPS stress tester*. `http://linux.die.net/man/1/siege`. Acessado em: 18 de Dezembro de 2015.

[Slowhttptest 2013] Slowhttptest (2013). *Slowhttptest tool*. `https://code.google.com/p/slowhttptest/`. Acessado em: 02 de Fevereiro de 2015.

[Slowloris 2013] Slowloris (2013). *Slowloris tool*. `http://ha.ckers.org/slowloris/`. Acessado em: 02 de Fevereiro de 2015.

[W3tech 2016] W3tech (2016). *Usage of web servers for website*. `http://w3techs.com/technologies/overview/web_server/all`. Acessado em: 18 de Agosto de 2016.

[Xie and Yu 2009] Xie, Y. and Yu, S.-Z. (2009). Monitoring the application-layer ddos attacks for popular websites. *Networking, IEEE/AcM Transactions on*, 17(1):15–25.

Privacidade em Dados Armazenados em Memória Compartilhada através de Espaços de Tuplas

Edson Floriano S. Junior[1] **, Eduardo Alchieri**[1] **, Diego F. Aranha**[2] **, Priscila Solis**[1]

[1] Departamento de Ciência da Computação – Universidade de Brasília – UnB
[2]Instituto de Computação – Universidade Estadual de Campinas – UNICAMP

Abstract. *Conceptually, tuple spaces are shared memory objects that provide operations to store and retrieve ordered sets of data, called tuples. Tuples stored in a tuple space are accessed by the contents of their fields, working as an associative memory. Although there are some proposals for secure tuple spaces, accessing tuples through field contents makes them susceptible to attacks that impair user and data privacy. To overcome this limitation, this work proposes some extensions to* DEPSPACE*, a tuple space that implements dependability properties, to improve data privacy. A deep analysis concerning security aspects of this system is presented, as well as the benefits of the proposed solutions.*

Resumo. *Um espaço de tuplas pode ser visto conceitualmente como um objeto de memória compartilhada que fornece operações para armazenar e recuperar conjuntos de dados ordenados chamados tuplas. As tuplas armazenadas em um espaço de tuplas são acessadas através do conteúdo de seus campos, funcionando assim como uma memória associativa. Embora existam algumas propostas que visam adicionar propriedades de segurança em espaços de tuplas, a necessidade de acesso através do conteúdo dos campos faz com que as mesmas sejam susceptíveis a ataques capazes de quebrar a privacidade dos dados e usuários do espaço. Visando contornar estes problemas, este trabalho propõe extensões ao* DEPSPACE*, um espaço de tuplas com propriedades de segurança, com o intuito de impedir que a privacidade dos dados e usuários seja afetada. Uma vasta análise acerca da segurança deste sistema é apresentada, bem como dos ganhos advindos com as soluções propostas.*

1. Introdução

A preocupação com aspectos de segurança tem ganhado espaço cada vez maior no projeto e desenvolvimento de aplicações distribuídas. Um sistema é dito seguro se satisfaz requisitos de integridade, disponibilidade e confidencialidade [Avizienis et al. 2004]. Intuitivamente, privacidade é entendida na perspectiva de uma entidade como a confidencialidade de suas informações sensíveis (dados e metadados) [Veríssimo 2016]. Esta entidade pode ser uma pessoa, uma organização, uma nação, etc. Como podemos observar, a privacidade está diretamente relacionada com a confidencialidade das informações.

Atualmente existem vários fatores que aumentam o risco à segurança das aplicações [Veríssimo 2016]: *(i)* o mundo está se tornando uma infraestrutura imensa, interconectada e interdependente; *(ii)* existem muitos dados correlacionados disponíveis; *(iii)* as entidades estão se expondo cada vez mais; e *(iv)* o número de vulnerabilidades de *software* está aumentando.

Em meio a este cenário, muitos sistemas visam manter a confidencialidade protegendo apenas os dados sigilosos, sem se preocupar com os dados não sigilosos relacionados. Entretanto, tendo em vista que ataques de inferência estatística, através da análise e correlação de dados de acesso público, muitas vezes conseguem obter informações mantidas sob sigilo [Naveed et al. 2015], torna-se interessante, sob o viés de segurança, proteger toda informação disponibilizada para uma aplicação.

Estes aspectos são particularmente relevantes quando consideramos dados compartilhados através de espaços de tuplas [Gelernter 1985, Bessani et al. 2008], onde o acesso é concretizado através do conteúdo dos campos das tuplas (memória associativa). Embora existam algumas propostas que visam adicionar propriedades de segurança neste modelo [De Nicola et al. 1998, Busi et al. 2003, Vitek et al. 2003, Bessani et al. 2008, Distler et al. 2015], a necessidade de acesso através do conteúdo dos campos faz com que as mesmas sejam susceptíveis a ataques capazes de quebrar a privacidade dos dados e usuários do espaço. Dentre as propostas existentes, o DEPSPACE [Bessani et al. 2008] é o sistema que provê um maior nível de segurança, empregando mecanismos tanto de controle de acesso quanto de criptografia. Este sistema sugere a classificação dos campos das tuplas em público, comparável ou privado. Para que o acesso a uma tupla seja possível, pelo menos um de seus campos precisa ser público ou comparável. Esta limitação torna o sistema susceptível a ataques de inferência estatística, conforme já discutido, ou ainda dificulta a sua utilização por aplicações, visto que as possibilidades de buscas por tuplas podem ser significativamente reduzidas para se preservar a segurança da informação.

Visando contornar estes problemas, este artigo propõe extensões ao DEPSPACE com o intuito de impedir que a privacidade dos dados e usuários seja afetada. Através do emprego de mecanismos de criptografia mais recentes e sofisticados, são preservadas tanto as propriedades de segurança do sistema quanto a flexibilidade das buscas no modelo tradicional de coordenação por espaço de tuplas. Desta forma, obtém-se um modelo em que a inclusão de segurança não afeta as possibilidades de buscas por tuplas. Adicionalmente, este trabalho faz uma vasta análise acerca da segurança do DEPSPACE, bem como dos aprimoramentos advindos com as soluções propostas.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 detalha o conceito de espaço de tuplas e apresenta o DEPSPACE, além de fazer uma análise acerca da segurança fornecida por este sistema. Na Seção 3 são discutidos esquemas criptográficos robustos que são utilizados na extensão proposta para o DEPSPACE, apresentada na Seção 4. Finalmente, os trabalhos relacionados relevantes são discutidos na Seção 5, enquanto que as conclusões e os trabalhos futuros são apresentados na Seção 6.

2. Espaço de Tuplas

Um espaço de tuplas pode ser visto (conceitualmente) como um objeto de memória compartilhada que fornece operações para armazenar e recuperar conjuntos de dados ordenados chamados de tuplas. Sendo assim, os processos de um sistema distribuído podem interagir através desta abstração de memória compartilhada. Uma tupla t é uma sequência ordenada de campos, onde um campo que contém um valor é dito definido. Um tupla onde todos os campos são definidos é chamada de entrada. Uma tupla $\overline{t}$ é chamada molde (ou *template*) se algum de seus campos não tem valor definido. Diz-se que uma tupla t e um molde $\overline{t}$ combinam se e somente se ambos têm o mesmo número de campos e todos os valores e tipos dos campos definidos em $\overline{t}$ são iguais aos valores e tipos dos campos cor-

respondentes em t. Por exemplo, uma tupla $\langle \text{SBRC}, 2017, \text{Belém} \rangle$ combina com o molde $\langle \text{SBRC}, 2017, * \rangle$ ('$*$' denota um campo sem definição do molde).

A coordenação por espaço de tuplas (generativa), introduzida pela linguagem de programação para sistemas paralelos LINDA [Gelernter 1985], suporta comunicações desacopladas no espaço (os processos não precisam conhecer as localizações uns dos outros) e no tempo (os processos não precisam estar ativos ao mesmo tempo). Além disso, este modelo de coordenação fornece algum poder de sincronização entre processos.

As manipulações realizadas no espaço de tuplas consistem em invocações de três operações básicas [Gelernter 1985]: $out(t)$ que adiciona a entrada t no espaço de tuplas; $in(\bar{t})$, que remove do espaço de tuplas uma tupla que combina com o molde $\bar{t}$; $rd(\bar{t})$, usada na leitura de uma tupla que combina com o molde $\bar{t}$, sem removê-la do espaço. As operações in e rd são bloqueantes, i.e., se não houver uma tupla que combine com o molde no espaço, o processo fica bloqueado até que uma esteja disponível. Uma extensão comum a este modelo é a inclusão de variantes não bloqueantes das operações de leitura, denominadas inp e rdp. Estas operações funcionam exatamente como as anteriores, a não ser pelo fato de retornarem mesmo não havendo uma tupla que combine com o molde usado (indicando esta inexistência). Note que, de acordo com as definições anteriores, o espaço de tuplas funciona como uma memória associativa: os dados são acessados a partir de seu conteúdo, e não através de seu endereço.

2.1. DEPSPACE: Um Sistema de Coordenação Tolerante a Falhas Bizantinas

Segurança é uma característica fundamental de sistemas confiáveis [Avizienis et al. 2004]. No contexto de um espaço de tuplas, os seguintes atributos são necessários: *confiabilidade* (as operações realizadas no espaço de tuplas fazem com que seu estado se modifique de acordo com a especificação), *disponibilidade* (o espaço de tuplas sempre está pronto para executar as operações requisitadas por partes autorizadas), *integridade* (nenhuma alteração imprópria no estado de um espaço de tuplas pode ocorrer), *confidencialidade* (o conteúdo dos campos das tuplas não pode ser revelado a partes não autorizadas). O DEPSPACE [Bessani et al. 2008] é a implementação de um espaço de tuplas que busca satisfazer estas propriedades por meio de diversas camadas, cada uma responsável pela concretização de uma funcionalidade diferente.

2.1.1. Camadas do DEPSPACE

Esta seção apresenta as camadas do DEPSPACE [Bessani et al. 2008], dando ênfase à camada de confidencialidade, que é responsável pelos aspectos discutidos neste trabalho.

Replicação. Para manter a consistência do espaço de tuplas, o DEPSPACE utiliza replicação Máquina de Estados [Schneider 1990, Castro and Liskov 2002]. Este mecanismo está relacionado principalmente com as propriedades de disponibilidade e confiabilidade, pois para o número n de réplicas no sistema, garante que o espaço de tuplas executa as operações a ele endereçadas seguindo sua especificação, mesmo que até $f = (n-1)/3$ réplicas sejam maliciosas (as réplicas corretas *mascaram* o comportamento das maliciosas). Através deste protocolo, as réplicas corretas executam a mesma sequência de operações e retornam os mesmos valores, evoluindo de forma sincronizada.

Confidencialidade. Conforme comentado, as tuplas são mantidas replicadas em um conjunto de servidores no DEPSPACE. Sendo assim, a preservação da propriedade de confidencialidade não pode ser atribuída a um único servidor, pois até f deles podem falhar e revelar o conteúdo das tuplas a partes não autorizadas.

Desta forma, a confidencialidade é implementada através do uso de um $(n,f+1)$ – esquema de compartilhamento de segredo publicamente verificável (*publicly verifiable secret sharing* – PVSS) [Schoenmakers 1999]. Os clientes, que são os distribuidores deste esquema, cifram as tuplas com um segredo por eles gerado. Após isso, geram um conjunto de n fragmentos (*shares*) deste segredo. Um segredo pode ser remontado apenas com a combinação de $f+1$ *shares*, o que torna impossível que um conluio de até f servidores faltosos revele o conteúdo de uma tupla. Este esquema utiliza um *fingerprint* da tupla para implementar a comparação entre tuplas e moldes, o qual é computado de acordo com o tipo dos campos escolhidos para a tupla, os quais podem ser:

- **Público (PU)**, o próprio valor do campo é o *fingerprint*, i.e, nenhum método criptográfico é aplicado ao conteúdo do campo que fica exposto;
- **Comparável (CO)**, um *hash* do valor do campo é o *fingerprint* (para isso utiliza uma função de *hash* resistente a colisões), possibilitando a execução de buscas (comparações); e
- **Privado (PR)**, um símbolo especial é o *fingerprint*, i.e., o conteúdo deste campo é mantido cifrado sem qualquer possibilidade de realização de comparações ou acesso aos dados.

Como não é possível enviar diferentes versões de uma requisição para diferentes servidores (contendo apenas seu *share* do segredo usado para cifrar a tupla), o cliente deve cifrar cada um dos *shares* com uma chave secreta compartilhada com o servidor que vai armazenar esse *share*. Deste modo, cada servidor terá acesso apenas ao seu *share* (um servidor faltoso não têm acesso a todos os *shares* para remontar e revelar a tupla). Assim, nas requisições de inserção de tuplas, o cliente envia aos servidores a tupla cifrada, os *shares* cifrados, as provas de que estes *shares* são válidos e o *fingerprint* da tupla.

Para acessar uma tupla, o cliente envia o *fingerprint* do molde e espera pelas respostas dos servidores. A resposta de cada servidor contém o *fingerprint* da tupla (que combina com o *fingerprint* do molde), a tupla cifrada e o *share* armazenado por este servidor[1] (juntamente com a prova de sua validade). O cliente decifra os *shares*, verifica suas validades e combina $(f+1)$ deles para obter o segredo e decifrar a tupla. Note que um cliente malicioso pode inserir uma tupla e informar um *fingerprint* que não corresponde ao *fingerprint* da tupla. Deste modo, após obter a tupla, o cliente deve verificar se a tupla corresponde ao *fingerprint*. Caso isso não aconteça, o cliente precisa eliminar esta tupla do espaço (se ainda não eliminou) e re-executar a operação. A eliminação de tuplas inválidas é realizada em dois passos: *(1)* o cliente envia todas as respostas recebidas para os servidores como prova que esta tupla é inválida (para isso, os servidores devem assinar as respostas a estas requisições); e *(2)* os servidores verificam a autenticidade das respostas e, se a tupla realmente é inválida, removem-na de seus espaços locais.

Vale destacar que pela forma como o *fingerprint* da tupla é definido, é necessário que cada tupla tenha campos públicos e/ou comparáveis de acordo com as buscas que

[1]O *share* é cifrado com a chave secreta compartilhada com o cliente para evitar captura das respostas.

serão realizadas (moldes utilizados nas buscas), sendo impossível suportar uma tupla apenas com campos privados pois nenhuma busca seria possível, i.e., os campos privados não podem ser utilizados para verificar se uma tupla e um molde combinam e são sempre usados como campos indefinidos no molde. Esta limitação traz pelo menos duas consequências. Por um lado, uma tupla com muitos campos privados faz com que a busca fique muito restrita e sem refinamento adequado, perdendo-se a flexibilidade na escolha de moldes, pois um molde com muitos campos indefinidos impossibilita uma busca mais refinada e limita o seu uso pelas aplicações. Por outro lado, uma tupla com muitos campos (dados) públicos e/ou comparáveis está susceptível a ataques, como veremos adiante.

Controle de Acesso O controle de acesso é um mecanismo fundamental para manutenção da integridade e confidencialidade das informações (tuplas) armazenadas no DEPSPACE, pois previne que clientes não autorizados obtenham acesso as tuplas, além de impedir que clientes faltosos saturem o espaço de tuplas enviando uma grande quantidade de tuplas. Atualmente, o DEPSPACE implementa controle de acesso de duas formas:

- **Baseado em credenciais:** para cada tupla inserida no DEPSPACE pode-se definir quais são as credenciais necessárias para acessá-la, tanto para leitura quanto para remoção (acesso em nível de tuplas). Estas credenciais são definidas pelo processo que insere a tupla. Também é possível definir, quando o espaço de tuplas é criado, quais são as credenciais necessárias para inserir uma tupla no espaço (acesso em nível de espaço). A implementação desta funcionalidade é realizada através da associação de listas de controle de acesso a cada espaço e tupla.
- **Políticas de granularidade fina:** o DEPSPACE suporta a definição de políticas de acesso de granularidade fina [Bessani et al. 2006], que devem ser especificadas no momento da criação do espaço de tuplas. Estas políticas controlam o acesso ao espaço considerando três parâmetros: o identificador do cliente, a operação que será executada (juntamente com seus argumentos) e o estado do espaço.

2.1.2. Análise de Segurança

Apresentaremos a seguir uma rápida explanação sobre algumas definições de segurança. Segundo [Menezes et al. 1996], os ataques aos esquemas criptográficos buscam obter o texto claro ou a chave de decifração através dos seguintes métodos:

- *Ciphertext-only attack (COA)*: Neste tipo de ataque, um adversário pode obter a chave de decifração ou o texto claro somente de posse do texto cifrado. Este é o tipo de ataque mais fraco e, portanto, um sistema vulnerável a este tipo de ataque não tem qualquer tipo de segurança.
- *Know-plaintext attack (KPA)* ou Ataque de texto claro conhecido: Neste tipo de ataque o adversário tem ao seu dispor uma significativa quantidade de textos claros e seus correspondentes textos cifrados e através desta comparação tenta obter a chave de decifração ou decifrar outros textos cifrados.
- *Chosen-plaintext attack (CPA)* ou ataque de texto claro escolhido: Neste tipo de ataque o adversário escolhe um texto claro e lhe é fornecido o texto cifrado correspondente, ele usa então a análise desta correlação para obter o texto claro correspondente a outro texto cifrado.

- *Adaptative chosen-plaintext attack (CPA2)*: Semelhante ao anterior, porém o atacante pode escolher novos textos claros dependendo da resposta recebida.
- *Chosen ciphertext attack (CCA)* ou ataque de texto cifrado escolhido: Neste tipo de ataque, o adversário escolhe um texto cifrado e lhe é fornecido (sem acesso à chave de decifração) o texto claro correspondente, ele então usa a análise desta correlação para obter o texto claro correspondente a outro texto cifrado.
- *Adaptative chosen-ciphertext attack (CCA2)*: Semelhante ao anterior, porém o atacante pode escolher novos textos cifrados dependendo da resposta recebida. Este ataque é considerado muito forte e de implementação mais difícil.

Os ataques citados estão em ordem de complexidade, de modo que um sistema vulnerável a um ataque mais fraco será classificado em um nível de segurança inferior, mesmo que resista a um ataque mais forte. Apesar de serem estes os principais ataques considerados pela literatura, uma infinidade de outros ataques pode ser possível dependendo das características do sistema. Por exemplo, [Naveed et al. 2015] mostram que é possível realizar os chamados *Ataques de inferência* por meio da correlação dos textos cifrados com informações adicionais publicamente disponíveis. Neste caso, havendo uma correlação forte entre estes dados cifrados e os públicos, os textos claros podem ser recuperados com grande acurácia. Em seu trabalho com bases de dados cifradas de hospitais, mais de 60% dos dados cifrados deterministicamente (Seção 3.1), como sexo, raça e risco de mortalidade, foram descobertos em 60% dos hospitais, enquanto que mais de 80% dos dados cifrados com preservação de ordem (Seção 3.2), como idade e nível de severidade de doença, foram recuperados em 95% dos hospitais.

Como na prática é impossível alcançar total segurança contra estes ataques para todos os adversários *matematicamente* possíveis, faz-se necessário um enfraquecimento da definição de segurança, levando em consideração apenas os adversários *computacionalmente* possíveis. Sob essa ótica, define-se informalmente como *semanticamente seguro* um sistema que, com alta probabilidade, é capaz de resistir aos ataques realizados por qualquer adversário computacionalmente eficiente [Boneh and Shoup 2015]. Baseados nas definições formais de [Bellare et al. 1998], definimos informalmente que para todo adversário eficiente **A**, uma cifra $\mathcal{E} = (E, D)$ definida sobre $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ pode oferecer:

- **Indistinguibilidade contra *ataques de texto claro escolhido (IND-CPA)*** se, para qualquer tentativa $i = 1, 2, ...q$, dadas duas mensagens $m_{i0}, m_{i1} \in \mathcal{M}$ de mesmo tamanho escolhidas por **A** e submetidas a um oráculo que responde com a cifra $c_i = E(k, m_{ib}) \in \mathcal{C}$ para alguma chave k selecionada aleatoriamente em $\mathcal{K}$ e $b \in \{0, 1\}$, a probabilidade de que **A** possa distinguir se $c_i = E(k, m_{i0})$ ou $c_i = E(k, m_{i1})$ é desprezível.
- ***Indistinguibilidade contra ataques de texto cifrado escolhido (IND-CCA)*** se, para as mesmas condições do item IND-CPA, o adversário $\mathcal{A}$ ainda obtiver acesso a um oráculo que dado um texto cifrado $c_i \notin \{c_1, ..., c_{i-1}\}$ responde com o **texto claro** $m_i = D(k, c_i)$ correspondente e da mesma forma a probabilidade de que $\mathcal{A}$ possa distinguir se $c_i = E(k, m_{i0})$ ou $c_i = E(k, m_{i1})$ é desprezível. Neste caso, $\mathcal{A}$ pode fazer quantas requisições quiser ao oráculo de decifração, porém somente até receber o criptograma desafio do oráculo de cifração.
- ***Indistinguibilidade contra ataques adaptativos de texto cifrado escolhido (IND-CCA2)*** se, além do estabelecido no item IND-CCA, o adversário puder continuar a usar o oráculo de decifração mesmo após receber o criptograma desafio, tendo como única restrição não poder submeter este criptograma para decifração.

Adicionalmente, temos as seguintes flexibilizações de IND-CPA para cifras determinísticas e de ordem:

- **Indistinguibilidade contra *Ataque de texto claro distinto escolhido (IND-DCPA)*** se a cifra $\mathcal{E}$ for determinística e se, para qualquer tentativa $i = 1, 2, ..., q$, dadas duas mensagens $m_{i0}, m_{i1} \in \mathcal{M}$ de mesmo tamanho escolhidas por **A**, distintas para cada tentativa, isto é, $\forall i, j \in \{1, 2, ..., q\}, m_{i0} \neq m_{j0}$ e $m_{i1} \neq m_{j1}$, submetidas a um oráculo que responde com a cifra $c_i = E(k, m_{ib}) \in \mathcal{C}$ para alguma chave k selecionada aleatoriamente em $\mathcal{K}$ e $b \in \{0, 1\}$, a probabilidade de que **A** possa distinguir se $c_i = E(k, m_{i0})$ ou $c_i = E(k, m_{i1})$ é desprezível [Bellare et al. 2004].
- **Indistinguibilidade contra *Ataques de texto claro ordenado escolhido (IND-OCPA)*** se a cifra $\mathcal{E}$ preservar a ordem dos textos claros e se, para qualquer tentativa $i = 1, 2, ...q$, dadas duas mensagens $m_{i0}, m_{i1} \in \mathcal{M}$ de mesmo tamanho escolhidas por **A** e submetidas sempre na mesma ordem (isto é, $m_{i0} < m_{j0} \iff m_{i1} < m_{j1}$ para todo $1 \leq i, j \leq q$) a um oráculo que responde com a cifra $c_i = E(k, m_{ib}) \in \mathcal{C}$ para alguma chave k selecionada aleatoriamente em $\mathcal{K}$ e $b \in \{0, 1\}$, a probabilidade de que **A** possa distinguir se $c_i = E(k, m_{i0})$ ou $c_i = E(k, m_{i1})$ é desprezível [Boldyreva et al. 2012].

Com isso podemos perceber que o sistema DEPSPACE apresenta algumas vulnerabilidades que devem ser pontuadas. Os campos comparáveis, apesar de trazerem ao sistema a grande vantagem de permitir a seleção de tuplas sem a necessidade de decifrá-las, por utilizar funções de *hash* são vulneráveis a ataques de pré-imagem, pois um adversário sempre pode obter qualquer quantidade desejada de entradas e suas respectivas saídas simplesmente calculando seus *hashes*. Este ataque assemelha-se ao ataque de texto claro conhecido, exceto pelo fato de que neste caso não há uma função de cifração ou decifração. Desta forma, se o conjunto de valores que um campo como este pode assumir for pequeno e conhecido, o atacante pode calcular os *hashes* de todos os valores possíveis, obtendo assim os textos claros correspondentes. Somado a isso, a existência de campos públicos pode proporcionar a um atacante a possibilidade de correlação entre os dados cifrados do sistema e uma base pública favorecendo os ataques de inferência.

Além disso, para prover o uso de criptografia sobre os dados armazenados nas tuplas, [Bessani et al. 2008] propõem deixar partes das chaves de decifração distribuídas entre os servidores, de modo que para recuperar a informação, um cliente deve receber um número mínimo de respostas $(f + 1)$ que possibilitam a composição da chave por completo. Porém, em um ambiente não confiável, um número $q \geq f + 1$ de servidores maliciosos poderiam entrar em conluio, juntar suas partes e recuperar as chaves, conseguindo acesso total ao conteúdo das tuplas, comprometendo toda a segurança do sistema.

3. Esquemas Criptográficos mais Robustos

Na busca por solucionar as vulnerabilidades e limitações citadas, apresentamos alguns modelos que permitem realizar buscas ou computações sobre dados cifrados e propomos como podem ser adaptados para o DEPSPACE. Baseados nas características do DEPSPACE, buscamos os esquemas de criptografia que melhor se encaixam em nossas necessidades em meio às muitas propostas existentes na literatura.

3.1. Cifras Determinísticas e Probabilísticas

Uma cifra $\mathcal{E} = (E, D)$ definida sobre $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ é dita *probabilística* se para entradas fixas de chave $k \in \mathcal{K}$ e mensagem $m \in \mathcal{M}$ da função de cifração $E : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$,

a saída $c = E(k, m)$ pode assumir valores diferentes. Caso contrário, a cifra será dita *determinística* [Boneh and Shoup 2015].

Cifras determinísticas possuem a característica de vazar igualdade de textos claros quando cifrados sob a mesma chave, ou seja, $m_0 = m_1 \iff E(k, m_0) = E(k, m_1)$. Essa característica pode ser utilizada para realizar buscas por igualdade em dados cifrados, como mostrado em [Popa et al. 2011, Alves and Aranha 2016]. Este tipo de cifra claramente não alcança segurança contra o ataque de texto claro escolhido, visto que um adversário pode enviar inicialmente ao oráculo duas cópias da mesma mensagem m_0 e receber portanto $c_0 = E(k, m_0)$, logo em seguida, envia as mensagens m_0 e m_1, recebendo em resposta $c_b = E(k, m_b)$ com $b \in \{0, 1\}$. Basta então comparar c_0 e c_b para ter a certeza de que se trata de $E(k, m_0)$ ou $E(k, m_1)$.

Já as cifras probabilísticas (ou aleatorizadas), possuem a característica de resistirem a ataques mais fortes, podendo atingir o nível de segurança IND-CCA2 quando combinadas com primitivas criptográficas de autenticação.

3.2. Order-Preserving Encryption - OPE

Boldyreva [et. al] propõe em [Boldyreva et al. 2012] um esquema criptográfico simétrico que preserva a relação de ordem, ou seja, para todo i e j e para toda chave k, $E(k, i) < E(k, j) \iff i < j$. Uma implementação com esta característica permite o vazamento de informações de ordem entre os textos cifrados, não satisfazendo portanto à premissa de IND-CPA. Admitindo que esta premissa não é alcançável por um algoritmo determinístico com tal propriedade, mesmo ante a uma flexibilização da premissa de segurança (IND-OCPA), os autores propõem utilizar funções e geradores pseudo-aleatórios (PRF's e PRG's) e provam que este desfruta de um nível flexibilizado porém ainda forte de segurança por eles chamado de *pseudorandom order-preserving function under chosen-ciphertext attack* (POPF-CCA), ou seja, uma função pseudo-aleatória com preservação de ordem que resiste ao ataque de texto cifrado escolhido.

Entretanto, esta definição de segurança não especifica quais dados, além da ordem, podem vazar ao se utilizar tal cifra. Em [Boneh et al. 2014] é proposta *Order Revealing Encryption (ORE)*, uma construção que minimiza a quantidade de dados vazados e portanto possui um nível mais robusto de segurança, porém inviável em termos de desempenho. Logo depois, [Chenette et al. 2015] propõem um algoritmo prático de ORE que atinge o nível de segurança IND-OCPA, porém com o vazamento do primeiro *bit* que diferencia os valores comparados. Posteriormente, [Lewi and Wu 2016] apresentam um algoritmo de ORE resistente aos ataques de inferência e que funciona muito bem em aplicações de Banco de Dados cifrados, como pode ser visto em [Alves and Aranha 2016].

3.3. Criptografia Homomórfica

A Criptografia Homomórfica [Morais and Dahab 2012] tem sido pesquisada como uma forma de se realizar computação em dados cifrados sem a necessidade de decifrá-los ou conhecer a chave de decifração. Isso se deve ao fato de que, em uma cifra homomórfica, dadas duas mensagens quaisquer m_1 e m_2, uma função de encriptação E parametrizada por uma chave k, tem-se que $E(k, m_1) \circ_c E(k, m_2) = E(k, m_1 \circ_m m_2)$, onde $\circ_m$ denota uma operação aritmética no domínio das mensagens e $\circ_c$ denota uma operação aritmética no domínio dos criptogramas. Entretanto, sistemas completamente homomórficos genéricos apresentam um desempenho impraticável para aplicações reais.

Em [Naehrig et al. 2011] os autores mostram que é possível evitar os problemas de desempenho dos sistemas completamente homomórficos, pois várias aplicações exigem apenas esquemas parcialmente homomórficos, suportando apenas alguns tipos de operações. Por apresentar desempenho muito superior, estes esquemas estão mais próximos de serem aplicáveis na prática. Os autores apresentam também uma implementação de um algoritmo cuja segurança se baseia no problema *Ring learning with errors (RLWE)* o qual suporta soma e multiplicação modulares com desempenho praticável. Paillier [Paillier 1999] e ElGamal exponencial [Gamal 1985] são exemplos de cifras parcialmente homomórficas eficientes e probabilísticas com nível de segurança IND-CPA.

4. Estendendo a Confidencialidade do DEPSPACE

Esta seção discute a proposta de incremento de segurança ao DEPSPACE através de uma arquitetura para compartilhamento de chaves e aplicação destes esquemas criptográficos ao protocolo. Como trata-se de um serviço voltado para sistemas distribuídos, o DEPSPACE pode ser modelado como grupos de processos clientes utilizando o espaço de tuplas, este por sua vez pode estar mantido em servidores não necessariamente confiáveis. Nossa primeira proposta consiste em retirar dos servidores qualquer informação sobre as chaves, pois como visto na Seção 2.1.2, um conluio de servidores maliciosos poderia recuperar as chaves por completo e comprometer toda a segurança do sistema.

Para evitar tal comprometimento, propomos que as chaves sejam de conhecimento apenas dos clientes, os quais devem previamente compartilhá-las utilizando um algoritmo de criptografia de chave pública com um mecanismo que disponha de proteção contra o ataque *Man-in-the-middle*, como sugerido em [Khader and Lai 2015]. Com isso podem co-existir grupos independentes de processos, cada um compartilhando seu próprio conjunto de chaves de acordo com a necessidade de compartilhamento de informações.

4.1. Protocolo

A análise apresentada na Seção 2.1.2 mostra que, devido à forma de como foi projetado e construído, o DEPSPACE está sujeito a uma série de ataques simples. Para contornar este problema, propomos a redução (ou eliminação) do uso de campos classificados como *públicos* ou *comparáveis* e a adoção da seguinte classificação:

- **Comparável Determinístico (CD)**, os quais serão cifrados utilizando um algoritmo determinístico de criptografia simétrica (Seção 3.1);
- **Ordenável (OR)**, os quais serão cifrados utilizando um algoritmo simétrico de criptografia com preservação da relação de ordem, como o OPE (Seção 3.2);
- **Operável (OP)**, os quais serão cifrados utilizando um algoritmo homomórfico ou parcialmente homomórfico (Seção 3.3).

Assim, para gerar o *fingerprint* $t_h = \langle h_1, ..., h_m \rangle$ de uma tupla $t = \langle f_1, ..., f_m \rangle$ e seu vetor de proteção $v_t = \langle v_1, ..., v_m \rangle$, a função $fingerprint(t, v_t) = t_h$ calcula cada $h_i = E_i(K_i, f_i)$, utilizando a cifra $\mathcal{E}_i$ e a chave compartilhada k_i de acordo com o tipo v_i. Caso $f_i = *$, então $h_i = *$, ou ainda se $v_i = PR$ então $h_i = PR$. O cliente então armazena no espaço de tuplas o *fingerprint* t_h e a tupla cifrada $t' = E_s(K_s, t)$ utilizando uma chave K_s e uma cifra simétrica $\mathcal{E}_s$. Com isso, continua garantido que a tupla t e um molde $\bar{t}$ combinam se t_h combina com $\bar{t}_h$. Então ao encontrar um *fingerprint* t_h que combina com um *fingerprint* $\bar{t}_h$ de um molde $\bar{t}$ utilizado para consulta, o servidor retorna a tupla cifrada t' que será decifrada pelo cliente utilizando a função $D_s(K_s, t') = t$ com a chave compartilhada K_s.

Devido aos motivos citados na Seção 2.1, aconselhamos fortemente a preferência pelo uso da classificação *comparável determinístico* proposta ao invés da *comparável*, pois utilizando uma cifra simétrica determinística ao invés do *hash*, um atacante precisaria de acesso à chave secreta para cifrar um texto, impossibilitando o ataque de pré-imagem. Entretanto, esta classificação deve ser utilizada com cuidado (Seção 2.1.2), pois como esta cifra revela a igualdade de textos claros, em um domínio muito pequeno utilizando poucas informações externas, o conteúdo destes campos pode ser facilmente descoberto. Por exemplo, ao utilizá-lo para cifrar um campo que contém o sexo em uma base que conhecidamente tem mais homens que mulheres, um atacante pode ver que existem apenas dois textos cifrados possíveis e concluir que aquele com mais ocorrências se refere ao sexo masculino. Portanto, este campo deve ser utilizado somente para campos que sirvam como índices, com grande quantidade de possíveis valores e que não sejam em si dados sensíveis, como ID's, endereços de *e-mail*, nomes de nós de processos, dentre outros.

Alguns algoritmos criptográficos utilizam modos de operação com Vetores de inicialização (IV) randomizados como forma de prover criptografia probabilística, tendo como opção de fixar estes IV's (em zero, por exemplo) como forma de prover criptografia determinística. Nestes casos, recomenda-se que, ao invés do uso do IV fixo, seja utilizado uma função pseudo-aleatória (PRF) sobre a mensagem utilizando uma chave k_1, produzindo uma saída pseudo-aleatória $r = F(k_1, m)$ e então utilizar r como IV da função de encriptação utilizando uma chave k_2, produzindo $c = E(k_2, m; r)$. Como a PRF gera a mesma saída para a mesma mensagem, o algoritmo continua determinístico, entretanto para mensagens diferentes a PRF gera saídas diferentes e portanto IV's diferentes para cada mensagem, alcançando o nível de segurança IND-DCPA [Boneh and Shoup 2015].

A classificação *ordenável*, por sua vez, possibilita uma gama de funcionalidades sobre as tuplas, como ordená-las por um campo deste tipo, realizar consultas de intervalo, selecionar máximos e mínimos, dentre outras. Da forma com que o DEPSPACE foi proposto, para se possibilitar este tipo de consulta, o campo precisaria estar classificado como *público*, ou caso os dados em questão sejam sensíveis, perde-se totalmente as funcionalidades classificando-se o campo como *privado*. Este tipo de funcionalidade é extremamente útil na execução de consultas mais complexas nos servidores [Distler et al. 2015]. Portanto, tanto o nível de segurança quanto o poder das buscas são aumentados. Entretanto há que se tomar bastante cuidado na escolha dos campos que utilizarão esta cifra, pois como demonstrado por [Naveed et al. 2015] um campo cifrado desta forma torna-se extremamente vulnerável aos ataques de inferência se todos os valores possíveis de um domínio estão presentes na coleção de dados. Por exemplo, se o campo referir-se à idade de pacientes em um hospital, que sabidamente tenha pacientes de todas as idades de 1 a 100 anos, os dados podem ser totalmente revelados devido a esta associação.

Como mostrado na Seção 3.2, o estado-da-arte deste tipo de algoritmo é o apresentado em [Lewi and Wu 2016]. Porém, por necessitar que parte dos dados sejam salvos no cliente para fazer as operações de comparação o algoritmo não se mostrou interessante para a aplicação no espaço de tuplas, pois neste caso os "clientes" são processos executando em um ambiente distribuído. Portanto, sugerimos que seja utilizado o algoritmo de OPE apresentado por [Boldyreva et al. 2012] e que a saída deste seja aplicada ao algoritmo prático de ORE da forma como sugerido por [Chenette et al. 2015]. Esta composição não causa grande impacto no desempenho e como provado pelos autores, faz com que o sistema atinja o nível de segurança IND-OCPA, vazando apenas o primeiro *bit*

que diferencia os valores comparados, neste caso da forma cifrada em OPE do número original ao invés do número propriamente dito.

Por último, a classificação de um campo como *operável* permite a computação sobre dados cifrados. Para isso, sugere-se o uso de uma cifra homomórfica ou parcialmente homomórfica (Seção 3.3), de acordo com a necessidade da aplicação. Um campo como este traria substancial incremento de funcionalidade ao DEPSPACE como a possibilidade de atualizar valores utilizados para sincronia de processos temporalmente desacoplados, sem revelar aos servidores o estado em que cada um se encontra. Para campos como este, que geralmente necessitam de apenas um tipo de operação (como soma/subtração), não se faz necessário o uso de uma cifra completamente homomórfica, uma cifra parcialmente homomórfica seria suficiente e não comprometeria de sobremaneira seu desempenho. Cabe ressaltar que mesmo utilizando algoritmos mais eficientes, um campo cifrado desta forma será o ponto crítico do desempenho do sistema, portanto propõe-se utilizá-lo somente quando extremamente necessário. Além disso, para estes campos, o cliente deve considerar o valor atualizado contido no *fingerprint* e não aquele da tupla.

Um dos fatos mais importantes a se considerar é que todas estas funcionalidades oferecidas por estas cifras podem ser utilizadas em uma busca ou operação sem que o servidor onde as tuplas estão armazenadas precise conhecer total ou parcialmente as chaves de decifração para os dados em questão, possibilitando que uma busca segura seja realizada mesmo se as tuplas estiverem armazenadas em um servidor não confiável.

Análise de Segurança. Como citado na Seção 2.1.2, uma cifra vulnerável a um ataque mais fraco será considerada como tendo um nível inferior de segurança, mesmo que seja resistente a um ataque mais forte. Tomando isso em conta, apresentamos na Tabela 1 o nível de segurança intrínseco a cifra utilizada em cada campo, ordenados do menor para o maior nível. Os níveis IND-OCPA e IND-DCPA são ambos flexibilizações do nível IND-CPA, porém além de igualdade, a cifra ordenável revela uma relação (ordem) entre textos cifrados diferentes, sendo portanto classificada em nível inferior ao IND-DCPA.

PU	CO	OR	CD	OP	PR
Inseguro	Pré-imagem/colisão	IND-OCPA	IND-DCPA	IND-CPA	IND-CCA2

Tabela 1. Nível de segurança dos campos no DEPSPACE

A segurança do sistema depende, portanto, do adequado uso de cada cifra, levando em consideração o tipo e o quão sensível é o dado em cada campo. Por exemplo, deve-se priorizar o uso destas cifras funcionais para campos utilizados para indexação e referência, evitando o uso em campos que contém dados mais críticos ou sensíveis. Mantendo todos os campos cifrados evita-se principalmente os ataques de inferência que de uma maneira computacionalmente mais simples causam grandes estragos mesmo a sistemas extremamente complexos.

5. Trabalhos Relacionados

Dentre os vários sistemas desenvolvidos com o intuito de introduzir segurança e/ou tolerância a falhas em espaços de tuplas, o DEPSPACE [Bessani et al. 2008] e a sua versão estendida para coordenação distribuída [Distler et al. 2015] são os únicos que consideram tanto mecanismos para tolerância a falhas (replicação) quanto para segurança (criptografia e mecanismos de controle de acesso). Alguns sistemas agregam apenas mecanismos

de replicação [Bakken and Schlichting 1995, Xu and Liskov 1989], enquanto outros utilizam apenas controle de acesso [Busi et al. 2003, De Nicola et al. 1998, Vitek et al. 2003]. Outros sistemas merecem destaque por introduzir suporte a tolerância a falhas através do conceito de transações [T. J. Lehman et al. 2001, GigaSpaces 2016, JavaSpaces 2016]. Estes trabalhos sobre segurança para espaço de tuplas têm um foco muito limitado, pois consideram apenas ataques simples (acesso inválido ou usam mecanismos criptográficos que não são fortes o suficiente para garantir a segurança da informação armazenada).

Mecanismos criptográficos mais robustos foram usados no contexto de banco de dados visando prover confidencialidade e proteção contra vazamento de informações através do processamento de consultas em bases de dados cifradas [Popa et al. 2011, Alves and Aranha 2016]. Nestes trabalhos, os autores apresentam estratégias de atribuição de esquemas diferentes de criptografia de acordo com a especificidade dos dados armazenados em cada campo e a categoria das consultas esperadas para tal campo, como igualdade, comparação, continência de palavras em textos, dentre outros. Em campos cifrados de forma homomórfica, por exemplo, podem ser efetuadas operações de UPDATE sem decifrar os dados. Porém, em campos de somente consulta, onde comparações utilizando inequações ($\leq$ ou $\geq$) são realizadas, cifras do tipo OPE e ORE são empregadas, apresentando um bom desempenho sem grande perda no nível de segurança. As soluções propostas nestes trabalhos possibilitam o emprego de mecanismos criptográficos mais robustos também no contexto de espaço de tuplas, visando mitigar problemas de segurança relacionados com as aplicações que utilizam este paradigma de programação.

6. Conclusão e Trabalhos Futuros

Este trabalho discutiu alguns problemas encontrados na implementação de segurança, principalmente privacidade, em dados compartilhados através de espaços de tuplas, os quais estão principalmente relacionados com o acesso às tuplas através do conteúdo dos seus campos. Para contorná-los, propõe-se a integração deste paradigma de programação com esquemas atuais e robustos de criptografia.

Como trabalhos futuros, pretende-se implementar tanto as extensões sugeridas ao DEPSPACE quanto uma aplicação que faça uso de toda sua potencialidade, possibilitando analisar o desempenho destes protocolos, embora os aspectos de segurança sejam mais relevantes. Outro trabalho futuro é o projeto e desenvolvimento de um protocolo para buscas eficientes em dados cifrados, de forma que o desempenho do sistema seja melhorado sem que as propriedades de segurança sejam afetadas.

Referências

Alves, P. G. M. R. and Aranha, D. F. (2016). A framework for searching encrypted databases. In *XVI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSEG 2016)*, pages 142–155.

Avizienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33.

Bakken, D. E. and Schlichting, R. D. (1995). Supporting Fault-Tolerant Parallel Programing in Linda. *IEEE Transactions on Parallel and Distributed Systems*, 6(3):287–302.

Bellare, M., Desai, A., Pointcheval, D., and Rogaway, P. (1998). Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology - CRYPTO*

'98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings, pages 26–45.

Bellare, M., Kohno, T., and Namprempre, C. (2004). Breaking and provably repairing the ssh authenticated encryption scheme: A case study of the encode-then-encrypt-and-mac paradigm. *ACM Trans. Inf. Syst. Secur.*, 7(2):206–241.

Bessani, A. N., Alchieri, E. P., Correia, M., and da Silva Fraga, J. (2008). DEPSPACE: A byzantine fault-tolerant coordination service. *European Conference on Computer Systems - EuroSys*, pages 163–176.

Bessani, A. N., Correia, M., Fraga, J. S., and Lung, L. C. (2006). Sharing memory between Byzantine processes using policy-enforced tuple spaces. In *Proceedings of 26th IEEE International Conference on Distributed Computing Systems - ICDCS 2006*.

Boldyreva, A., Chenette, N., Lee, Y., and O'Neill, A. (2012). Order-preserving symmetric encryption. Cryptology ePrint Archive, Report 2012/624. `http://eprint.iacr.org/2012/624`.

Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., and Zimmerman, J. (2014). Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. Cryptology ePrint Archive, Report 2014/834. `http://eprint.iacr.org/2014/834`.

Boneh, D. and Shoup, V. (2015). A graduate course in applied cryptography. `https://crypto.stanford.edu/~dabo/cryptobook/draft_0_2.pdf`. Acessado: 2016-11-30.

Busi, N., Gorrieri, R., Lucchi, R., and Zavattaro, G. (2003). SecSpaces: a Data-Driven Coordination Model for Environments Open to Untrusted Agents. In *Electronic Notes in Theoretical Computer Science*, volume 68.

Castro, M. and Liskov, B. (2002). Practical Byzantine fault-tolerance and proactive recovery. *ACM Transactions Computer Systems*, 20(4):398–461.

Chenette, N., Lewi, K., Weis, S. A., and Wu, D. J. (2015). Practical order-revealing encryption with limited leakage. Cryptology ePrint Archive, Report 2015/1125. `http://eprint.iacr.org/2015/1125`.

De Nicola, R., Ferrari, G. L., and Pugliese, R. (1998). KLAIM: A Kernel Language for Agents Interaction and Mobility. *IEEE Transactions on Software Engineering*, 24(5):315–330.

Distler, T., Bahn, C., Bessani, A., Fischer, F., and Junqueira, F. (2015). Extensible distributed coordination. In *Proceedings of the Tenth European Conference on Computer Systems*, EuroSys '15.

Gamal, T. E. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472.

Gelernter, D. (1985). Generative Communication in Linda. *ACM Transactions on Programing Languages and Systems*, 7(1):80–112.

GigaSpaces (2016). GigaSpaces Homepage. Disponível em `http://www.gigaspaces.com/`.

JavaSpaces (2016). JavaSpaces guide. Disponível em `http://www.oracle.com/technetwork/articles/java/javaspaces-140665.html`.

Khader, A. S. and Lai, D. (2015). Preventing man-in-the-middle attack in diffie-hellman key exchange protocol. In *22nd International Conference on Telecommunications, ICT 2015, Sydney, Australia, April 27-29, 2015*, pages 204–208.

Lewi, K. and Wu, D. J. (2016). Order-revealing encryption: New constructions, applications, and lower bounds. In *ACM Conference on Computer and Communications Security (ACM CCS)*.

Menezes, A. J., Vanstone, S. A., and Oorschot, P. C. V. (1996). *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition.

Morais, E. and Dahab, R. (2012). Encriptação homomórfica. In *Minicursos do XII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSEG 2012)*, pages 151–195.

Naehrig, M., Lauter, K., and Vaikuntanathan, V. (2011). Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 113–124, New York, NY, USA. ACM.

Naveed, M., Kamara, S., and Wright, C. V. (2015). Inference attacks on property-preserving encrypted databases. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 644–655.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99, pages 223–238, Berlin, Heidelberg. Springer-Verlag.

Popa, R. A., Redfield, C. M. S., Zeldovich, N., and Balakrishnan, H. (2011). CryptDB: Protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 85–100.

Schneider, F. B. (1990). Implementing fault-tolerant service using the state machine aproach: A tutorial. *ACM Computing Surveys*, 22(4):299–319.

Schoenmakers, B. (1999). A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology - CRYPTO'99*, pages 148–164.

T. J. Lehman et al. (2001). Hitting the distributed computing sweet spot with TSpaces. *Computer Networks*, 35(4):457–472.

Veríssimo, P. (2016). Dialogue on cyber policies between brazil and the eu: prospecting threats and opportunities of the cyberspace. Dialogue on Cyber Policies.

Vitek, J., Bryce, C., and Oriol, M. (2003). Coordination processes with Secure Spaces. *Science of Computer Programming*, 46(1-2):163–193.

Xu, A. and Liskov, B. (1989). A design for a fault-tolerant, distributed implementation of Linda. In *Proc. of the 19th Symposium on Fault-Tolerant Computing*, pages 199–206.

Trilha Principal do SBRC 2017
Sessão Técnica 25
Redes sem Fio II

FS-MAC: Uma Plataforma para a Flexibilização da Subcamada MAC em Redes Sem Fio

Jefferson R. S. Cordeiro, Esthefanie Lanza, Daniel F. Macedo, Luiz F. M. Vieira

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais

{jeff,esthefanie.lanza,damacedo,lfvieira}@dcc.ufmg.br

Abstract. *Current wireless networks are very dynamic and have a wide variety of applications with very different needs. Applications such as voice and video, for example, require low latency to ensure quality in the transmission, but tolerate reasonable error rates. On the other hand, web and message services have low error tolerance, however don't require low latency. This diversity raises the need for more flexible equipment as well as networks that adapt to the context of the network. This work make the medium access control more flexible, allowing more than one MAC protocol to be used on the network. This operates by activating each one in the situation where it is most efficient. FS-MAC is also extensible, allowing the addition of new protocols. The proposal was tested in a testbed, where we varied the load and the number of connected stations.*

Resumo. *As redes sem fio atuais são muito dinâmicas e abrigam uma grande diversidade de aplicações com necessidades bem diferentes. Aplicações de voz e vídeo, por exemplo, exigem uma latência baixa para que haja qualidade na transmissão, mas suportam taxas razoáveis de erro. Por outro lado, serviços de mensagem e web são pouco tolerantes a erros, mas não requerem baixa latência. Essa diversidade faz surgir a necessidade por equipamentos e redes mais flexíveis, que se adaptem às características do contexto. O presente trabalho flexibiliza o controle de acesso ao meio, permitindo que mais de um protocolo MAC possa ser utilizado na rede, colocando cada um em atividade na situação em que é mais eficiente. O FS-MAC*[1] *ainda é extensível, permitindo a adição de novos protocolos. A proposta foi testada em um testbed, onde variamos a carga e o número de estações conectadas.*

1. Introdução

As redes sem fio atuais são muito dinâmicas e suas características possibilitam a existência de uma grande diversidade de aplicações. Consideremos, por exemplo, uma rede de celular que promove a comunicação entre diversos dispositivos, entre eles, *smartphones*. Esses aparelhos possuem aplicações que executam chamadas de voz, chamadas de vídeo, exibição de páginas web, aplicativos de troca de mensagens, etc. As aplicações de chamada de voz e vídeo possuem uma tolerância razoável a erros, pois a perda de alguns pacotes não compromete o entendimento da mensagem por parte do usuário. Por outro lado, essas aplicações exigem uma latência pequena, pois no nível do usuário, atrasos grandes prejudicam muito este tipo de comunicação. Já as aplicações

[1]Código disponível em https://github.com/jeffRayneres/FS-MAC

web e de mensagem de texto por exemplo, suportam latências maiores. No entanto, a tolerância a erros nesses casos é pequena, pois a falta mínima de alguns dados pode comprometer a exibição e o entendimento do conteúdo. Assim, torna-se desejável que a rede adapte o seu comportamento de acordo com as necessidades das aplicações, alterando dinamicamente parâmetros de um protocolo, por exemplo.

Na subcamada MAC há também essa demanda por flexibilidade, pois os protocolos existentes não são eficientes em todos os contextos [Sharp et al. 1995, Rhee et al. 2008]. Isto ocorre porque é impossível que um único protocolo consiga atender a requisitos ortogonais como largura de banda, latência, consumo de energia, disponibilidade e segurança. Assim, percebemos uma necessidade por redes e dispositivos mais flexíveis que possam se adaptar ao contexto da comunicação para maximizar o aproveitamento dos recursos e atender corretamente aos requisitos das aplicações.

Uma solução adaptativa poderia empregar as diversas técnicas existentes na subcamada MAC para atender às demandas da aplicação. O MAC possui diversos algoritmos e técnicas que podem ser usados de acordo com a necessidade do projetista, por exemplo: i) acesso com ou sem contenção; ii) utilização ou não controle de fluxo; iii) emprego de *Automatic Repeat Request* (ARQ) ou *Forward Error Correction* (FEC); iv) priorização do acesso ao meio; v) reserva de recursos; vi) mecanismos de segurança e privacidade.

A literatura apresenta diversas soluções para a adaptatividade. Os protocolos híbridos [Sharp et al. 1995, Rhee et al. 2008, Hu et al. 2011] ajustam parte dos seus parâmetros de operação para um conjunto limitado de situações. Plataformas programáveis [Neufeld et al. 2005, Nychis et al. 2009, Tinnirello et al. 2012] permitem o desenvolvimento de novos protocolos MAC adaptados a cada rede ou serviço, mas em geral possuem uma expressibilidade limitada. Finalmente, podemos empregar plataformas de troca de protocolos [Doerr et al. 2005], onde a plataforma escolhe qual protocolo empregar dentre uma lista de protocolos disponíveis, baseado nas demandas da rede.

Este artigo apresenta uma arquitetura híbrida e com alta flexibilidade para a subcamada MAC, chamada FS-MAC. O FS-MAC troca dinamicamente o protocolo MAC em atividade na rede, de forma a manter operando aquele que for mais eficiente para o cenário apresentado. A arquitetura é extensível, permitindo a adição de novos protocolos MAC. A troca entre protocolos é regida por um conjunto de regras, que podem ser customizadas de acordo com as necessidades da aplicação e dos administradores da rede. Implementamos um protótipo sobre o padrão IEEE 802.15.4, onde o FS-MAC escolhe entre um protocolo MAC baseado em CSMA e outro baseado em TDMA.

Os resultados mostram que o FS-MAC funciona como esperado, pois o protótipo implementado é capaz de detectar que protocolo melhor se adapta às condições da rede e quando necessário, de forma coordenada, substitui corretamente o protocolo em atividade pelo mais adequado. Os resultados mostram também que devido às mensagens de controle necessárias para o seu funcionamento, o FS-MAC apresentando uma pequena queda de desempenho em relação aos protocolos isolados.

O restante do artigo está organizado da seguinte forma: a seção 2 discute os trabalhos relacionados ao tema tratado nesse artigo; a seção 3 apresenta a arquitetura FS-MAC; a seção 4 mostra a implementação de um protótipo baseado na arquitetura proposta; a seção 5 descreve os experimentos realizados para a avaliação da arquitetura e discute os

seus resultados e por fim a seção 6 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

Desde a criação de redes sem fio, diversos trabalhos foram publicados com a preocupação de melhorar o controle de acesso ao meio. Com a popularização dessas redes e a diversificação dos protocolos, várias pesquisas foram desenvolvidas no sentido de aumentar a flexibilidade e o desempenho da subcamada MAC.

Diversos trabalhos procuram otimizar os protocolos existentes. Os autores de [Puschmann et al. 2013] mostram estratégias que reduzem o parâmetro de *slot time*, mitigando os efeitos negativos da pouca capacidade de processamento de sinais. Já o OpenTDMF é inspirado no conceito de SDN [Yang et al. 2015]. O controlador associa um conjunto de *slots* de tempo e um nível de prioridade a cada fluxo. De acordo com o valor de prioridade, cada ponto de acesso determina ordens de transmissão em nível de pacote de seus fluxos locais. A abordagem reduz os efeitos de terminais expostos e escondidos e aumenta a justiça na rede. Apesar disso, a abordagem ainda apresenta um alto *overhead* quando há poucos dispositivos tentando transmitir.

Otimizar os protocolos existentes aumentam o seu desempenho. Entretanto, todo protocolo possui limitações em função das suas premissas básicas do modo de acesso ao meio. Assim, outra vertente de investigação prioriza os protocolos híbridos, em que diversos modos de acesso são incorporados em um único protocolo MAC.

Sharp et al. propuseram um protocolo híbrido, que alterna entre CSMA e TDMA [Sharp et al. 1995]. Apesar da contribuição, a solução foi implementada sobre um Sistema para demonstração de Rádio Pacotes (PRDS), um equipamento de difícil acesso fora de grandes laboratórios. Diversos protocolos híbridos foram propostos desde então. O padrão IEEE 802.11 suporta operação híbrida. O *Distributed Foundation Wireless Medium Access Control* é um protocolo híbrido implementado no IEEE 802.11 [Diepstraten and WCND-Utrecht 1993], onde a rede trabalha em um período com contenção (PCF) e outro livre de contenção (DCF). Entretanto, a proporção alocada para cada período é fixa.

O Z-MAC é um protocolo híbrido baseado em *slots* [Rhee et al. 2008]. O Z-MAC utiliza um algoritmo de escalonamento para a alocação dos *slots*. Assim, cada *slot* possui um *dono*, que tem prioridade sobre o seu uso. Quando o *slot* não é utilizado pelo dono, ele pode ser usado por outros nós. O uso de prioridades gera um efeito implícito de alternância entre CSMA e TDMA. O LA-MAC alterna entre os modos CSMA e HYBRID (baseado em TDMA) em função do nível de contenção da rede [Hu et al. 2011]. A métrica utilizada para essa decisão é a quantidade de perdas de pacotes em uma vizinhança de dois saltos. Quando em baixa contenção, todos os nós competem igualmente pelos *slots* (CSMA). Já no estado de alta contenção, os nós competem pelos *slots* com base em suas prioridades, que são baseadas na sua janela de contenção (modo HYBRID).

AGENT utiliza um protocolo baseado em contenção dentro de um protocolo livre de contenção: há um CSMA em operação dentro dos *slots* de tempo do TDMA [Myers et al. 2002]. Dessa forma, nos cenários onde a carga na rede é baixa, o protocolo tem eficiência similar ao CSMA, enquanto em cenários de maior congestionamento, se comporta similarmente ao TDMA. Apesar da flexibilidade, o protocolo se limita ao uso de uma implementação específica de TDMA e outra de CSMA, não deixando espaço para a inclusão de outros protocolos.

Apesar de serem uma evolução dos protocolos MAC clássicos, os protocolos híbridos também são imutáveis. Isso impede que o MAC acompanhe as mudanças nos requisitos das aplicações e na carga da rede.

Uma forma de flexibilizar o MAC é permitir a sua programabilidade a partir de um conjunto de instruções ou de APIs. O SoftMAC [Neufeld et al. 2005] e o FLAVIA [Tinnirello et al. 2012] permitem a reprogramação do MAC em placas de redes comerciais de baixo custo. O WiSHFUL é uma arquitetura que permite a programação de placas de diversos modelos, empregando uma linguagem comum [Ruckebusch et al. 2016]. Já em [Nychis et al. 2009] os autores identificam um conjunto de funções MAC importantes que devem ser implementadas mais próximas do rádio por razões de eficiência, e definem também uma API que permite ao *host* controlar essas funções, provendo a flexibilidade necessária para a implementação de novos protocolos MAC. Ambos os trabalhos focam na criação de novos protocolos MAC, e não na adaptação dinâmica. O *Overlay MAC Layer* (OML) adiciona uma nova camada sobre a subcamada MAC, que permite adaptar o controle de acesso ao meio a requisitos de roteamento e aplicação, por exemplo [Rao and Stoica 2005]. No entanto, ela está limitada à interface de comunicação entre a camada MAC e a camada de redes. Não é possível, por exemplo, realizar a verificação do meio, pois informações da camada física não estão disponíveis para os níveis superiores.

Uma segunda alternativa de flexibilização é o emprego de plataformas que trocam dinamicamente entre diversos protocolos. A programabilidade do MAC fornece somente algumas primitivas de alto nível por razões de desempenho, limitando as características que podem ser programadas. Enquanto isso, a troca dinâmica permite um maior grau de liberdade para os protocolos MAC.

A plataforma MultiMAC pode abrigar diversos protocolos, onde cada um é utilizado na situação mais adequada [Doerr et al. 2005]. O MultiMAC permite não só o uso de CSMA em situações de baixa contenção e TDMA em situações de alta contenção, mas prevê a inclusão de novos protocolos nesse conjunto. A troca entre protocolos é baseada em regras, que avaliam quando um dado protocolo deve entrar em operação. A avaliação emprega dois protocolos, sendo um com detecção de erros (CRC) e outro com correção de erros. Os pontos fracos do trabalho são a falta de um mecanismo de sincronização entre as estações, o que é importante para redes de larga escala, pois alguns protocolos podem ser conflitantes entre si, gerando problemas de desempenho se as estações não entram em consenso sobre o melhor protocolo a ser utilizado. Assim, neste trabalho propomos um mecanismo centralizado de decisão.

3. A Plataforma FS-MAC

A plataforma *Flexible System for Medium Access Control* (FS-MAC) é uma arquitetura que promove a flexibilização da subcamada MAC de redes sem fio. Esta arquitetura foi desenvolvida para permitir o uso de mais de um protocolo no controle de acesso ao meio, utilizando cada um na situação em que é mais eficiente. Para tanto, ela emprega regras baseadas em lógica *fuzzy*.

O FS-MAC possui módulos e um conjunto de protocolos conforme representado na Figura 1. O módulo de *Sensoriamento* reune dados sobre o contexto da rede e os repassa ao módulo de Decisão. A decisão no FS-MAC é centralizada, ocorrendo em um nó chamado *coordenador*. O objetivo do módulo de *Decisão* é analisar os dados recebidos,

decidir qual protocolo se adapta melhor às atuais condições da rede e repassar essa análise ao módulo de Troca. Finalmente, o módulo de *Troca* determina se a troca de protocolo é necessária, caso afirmativo, ele notifica os nós da decisão tomada e executa todos os procedimentos para realizar a substituição do protocolo em operação. Um exemplo em que pode ser empregada essa solução são as redes Wi-Fi com o ponto de acesso sendo o coordenador FS-MAC.

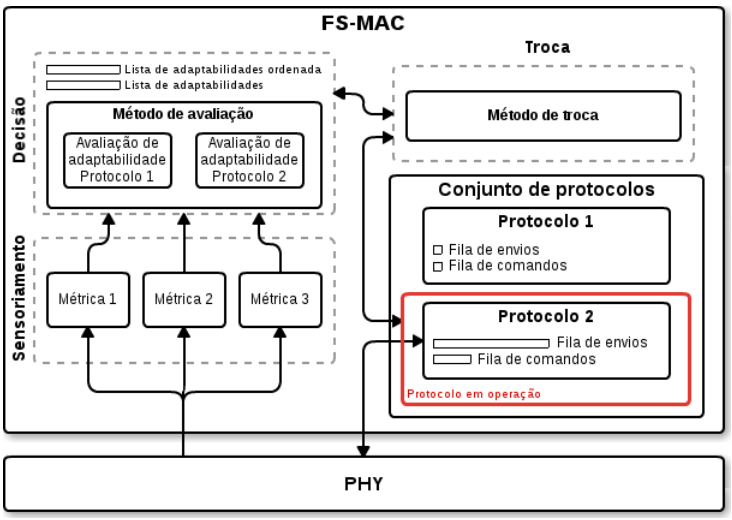


Figura 1. Arquitetura do FS-MAC.

A grande vantagem do FS-MAC sobre as soluções existentes é o alto grau de extensibilidade e de configuração. O módulo de Sensoriamento permite a adição de novas métricas a serem monitoradas, que poderão ser usadas no módulo de Decisão. Já o módulo de Decisão utiliza um conjunto de regras, que podem ser trocadas pelos administradores ou pelos usuários da rede, de forma que a escolha de protocolos seja adaptada às suas necessidades. Finalmente, o FS-MAC permite a adição de novos protocolos MAC, desde que esses implementem os métodos necessários para a operação do FS-MAC.

3.1. Módulo de Sensoriamento

O módulo de Sensoriamento colhe informações da rede que serão usadas para decidir qual protocolo MAC será utilizado. Este módulo é composto por unidades básicas de coleta e processamento, cada unidade é responsável por coletar dados da rede, processá-los e gerar o valor de uma métrica específica. Algumas métricas possíveis são: quantidade de estações se comunicando, volume de pacotes enviados, tipo de fluxo (contínuo ou rajadas) e quantidade de pacotes confirmados.

O FS-MAC suporta tanto sensoriamento centralizado quanto sensoriamento distribuído. No caso de sensoriamento centralizado, apenas o nó coordenador realiza as coletas. Um exemplo de métrica centralizada é o número de estações transmitindo. Quando o sensoriamento é distribuído, todas as estações recolhem dados individualmente e enviam o valor calculado da métrica ao módulo de Decisão do nó coordenador. Alguns exemplos de métricas distribuídas são taxa de entrega e latência. Neste caso, cada nó da rede envia os dados coletados para o coordenador, que os repassa para o módulo de Decisão.

Embora o sensoriamento centralizado seja mais simples, a decisão de manter a possibilidade de um sensoriamento distribuído se apoia no fato de que algumas métricas

não podem ser geradas centralizadamente, pois perdem precisão nesse caso. O valor de uma métrica como taxa de entrega, por exemplo, não pode ser calculado centralizadamente, pois a um nó não é possível saber isoladamente se outro recebeu a confirmação de seus pacotes. Outra métrica como nível de contenção da rede, quando calculada centralizadamente levando em consideração dados locais, pode não corresponder à realidade da rede, principalmente durante a execução de protocolos como CSMA/CA onde não há garantia de justiça.

Cada métrica é sensoreada a partir de um submódulo específico. Isto permite que novas métricas sejam adicionadas ao FS-MAC de acordo com os protocolos MAC empregados ou a partir das necessidades da rede.

3.2. Módulo de Decisão

A decisão de qual protocolo MAC mais se adapta às condições da rede ocorre em um nó central, o coordenador. O módulo de Decisão recebe as informações do Sensoriamento e calcula a adaptabilidade de cada protocolo ao contexto atual da rede. Para tanto, o módulo precisa conhecer as características dos protocolos, de forma a identificar onde eles apresentam maior eficiência e as situações onde não possuem bom desempenho.

A representação das características dos protocolos é feita por regras, que são expressadas em lógica *fuzzy*. O administrador da rede pode adaptar o funcionamento do FS-MAC ao mudar as regras de cada protocolo, por exemplo, priorizando desempenho, consumo de energia, confiabilidade ou outras características dos mesmos.

O motor de decisão avalia a adaptabilidade de cada protocolo, ou seja, o quão adequado ele é para o estado atual da rede. Assim, o módulo de Decisão gera uma lista de adaptabilidade, que é repassada para o módulo de Troca. Este desenho garante o baixo acoplamento entre os dois módulos, pois o módulo de Decisão não precisa saber que protocolo está em operação no momento.

3.3. Módulo de Troca

O módulo de Troca verifica se é necessário modificar o protocolo MAC empregado no momento. Esta operação acontece em três passos:

1. A lista de adaptabilidades é analisada, se o protocolo em operação é diferente daquele com maior grau de adaptabilidade na lista, os dois protocolos são comparados. Para que a troca seja viável, é preciso que a diferença entre a adaptabilidade dos dois protocolos seja maior que um certo limiar. A definição desse limiar deve garantir que o custo operacional da troca possa ser compensado pela melhora de desempenho na rede. Caso a diferença seja maior, o protocolo deve ser efetivamente trocado e os passos seguintes são executados, caso contrário, mantem-se o protocolo atual.
2. Depois de concluído que deve haver a troca, o nó coordenador comunica imediatamente a todos os nós qual deve ser o novo protocolo a entrar em operação, enviando uma mensagem à rede.
3. Internamente, cada nó da rede substitui o protocolo em operação pelo novo protocolo definido.

A troca de protocolos é notificada ao protocolo atual, para que este possa terminar as suas operações em um estado consistente. Feito isso, o novo protocolo é ativado e

recebe o controle da fila de pacotes a serem enviados. Esta fila contém todos os pacotes que ainda não foram enviados e confirmados, na ordem em que aparecem na estrutura.

A efetivação da troca pode ser implementada de duas formas diferentes, dependendo de como a plataforma irá tratar a coexistência de protocolos distintos:

- Conduzir uma troca distribuída atomicamente, ou seja, enquanto houver pelo menos um nó operando no protocolo antigo, o novo não entra em operação.
- O coordenador avisa a todos os nós periodicamente qual protocolo deverá estar em atividade, minimizando assim o tempo de operação com protocolos diferentes.

Na primeira opção, nunca haverá protocolos diferentes operando ao mesmo tempo. No entanto, esta abordagem é mais complexa e lenta, e dificulta a entrada e saída de nós na rede. Assim, escolhemos a segunda abordagem. A sua desvantagem é que o uso de protocolos diferentes pode degradar o desempenho da rede. Entretanto, o aviso periódico do coordenador faz com que a quantidade de nós empregando o protocolo antigo diminua com o tempo.

3.4. Conjunto de Protocolos MAC

É o conjunto de protocolos MAC implementados. Para acrescentar novos protocolos a esse conjunto, devemos implementar duas interfaces: *Interface de troca* e *Interface de comunicação externa*. A interface de troca possibilita que o FS-MAC interrompa um protocolo em funcionamento e ative outro protocolo. Já a interface de comunicação externa possibilita ao FS-MAC solicitar o envio de mensagens de controle, tais como as mensagens de envio de dados de sensoriamento e as mensagens de troca de protocolo. Além disso, o gerente da rede deve escrever as regras de troca para o protocolo e instalá-las no módulo de decisão. Essas regras devem descrever o grau de adaptabilidade do protocolo, para que o protocolo mais adequado seja escolhido.

4. Implementação do FS-MAC em SDR

Esta seção descreve uma implementação do FS-MAC que permite a troca entre um protocolo CSMA/CA e um protocolo TDMA. Na implementação utilizamos a tecnologia SDR [Silva et al. 2016] através do *framework* Ettus USRP com o apoio da plataforma de desenvolvimento GNU Radio. Como base para a implementação, utilizando a pilha ZigBee desenvolvida por [Schmid et al. 2016]. Como a implementação de ZigBee selecionada se limitava à camada física e ao encapsulamento dos quadros, desenvolvemos novos módulos que implementam o protocolo CSMA/CA do ZigBee. A partir dessa pilha, substituímos o bloco da camada MAC pelo conjunto de blocos que compõem a plataforma FS-MAC, construídos por nós. O código foi desenvolvido parte em C++ e parte em Python.

O FS-MAC é uma arquitetura genérica, que permite o sensoriamento de várias métricas bem como protocolos e modelos de decisão diferentes, assim iremos detalhar a seguir as características que implementamos para a avaliação experimental que será apresentada na próxima seção.

4.1. Protocolos MAC Implementados

Protocolo CSMA/CA: foi implementado um CSMA/CA com back-off exponencial binário e confirmação de envio de mensagens, como ocorre no ZigBee.

Protocolo TDMA: O TDMA implementado tem como principais características: a alocação de *slots* de tempo distintos para a comunicação de cada nó e a confirmação de mensagens. Existem dois tipos diferentes de nós, o nó normal, que apenas se comunica com outros nós e o nó coordenador, que além de se comunicar normalmente, também gerencia a distribuição de *slots* de tempo para a troca de mensagens. O nó coordenador é fixo durante todo o tempo de execução, e é determinado por uma variável no GNU Radio.

O tempo é dividido em intervalos chamados *superquadros*, neste intervalo, todos os nós têm a chance de se comunicar com outros nós transmitindo dados e recebendo confirmações. O *superquadro* e suas subdivisões estão representados na Figura 2. O superquadro é formado por dois períodos: o **período de alocação do canal**, em que o nó coordenador envia uma mensagem *broadcast* informando a ordem em que os nós devem requisitar a alocação de um *slot* de comunicação. Esta mensagem também serve como marco de início do superquadro, sincronizando o processo entre todos os nós. O **período de comunicação** define uma quantidade de *slots* igual à quantidade de nós que requisitaram o canal. Usando o mesmo mecanismo da fase de alocação, cada nó espera o momento correto para se comunicar. Quando chega esse momento, ele envia a mensagem que deseja e aguarda a confirmação.

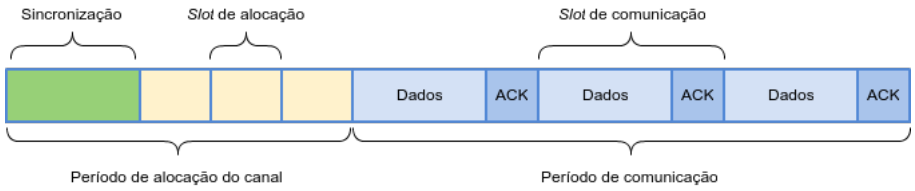


Figura 2. Formato do superquadro do TDMA implementado.

4.2. Métricas de sensoriamento

Para a fase de Sensoriamento, foram implementadas duas métricas, *Quantidade de transmissores* e *Latência*.

A métrica de quantidade de transmissores é calculada de forma centralizada pelo nó coordenador. Todos os pacotes vindos da camada física passam pelo bloco dessa métrica, que armazena os endereços únicos encontrados na rede. Essa métrica é amostrada a cada 5 segundos, ou seja, ao fim de cada intervalo o valor da métrica é enviado ao módulo de Decisão, o conjunto de endereços únicos é zerado e o processo de armazenamento desses endereços é reiniciado.

Já os dados da métrica de latência são colhidos de forma distribuída, ou seja, cada nó reúne as informações locais referentes a latência de entrega dos pacotes e as envia para o nó coordenador para que seja gerado o valor da métrica que será enviado ao módulo de Decisão. A latência é calculada como o tempo entre o envio do quadro de dados e a recepção de uma confirmação de recepção, de forma a contabilizar a quantidade de retransmissões necessárias para que o quadro seja recebido com sucesso. Os nós enviam os dados coletados para o coordenador a cada 10 segundos.

4.3. Modelo de decisão

O modelo de decisão construído emprega lógica *fuzzy*, e é modelado como descrevemos a seguir.

Variáveis linguísticas: O modelo considera três variáveis linguísticas, que são (i) Latência média dos pacotes (*LM*), $ii)$ número de transmissores (*NT*) e $iii)$ Adaptabilidade do protocolo (*ADP*). Todas elas aceitam os termos nebulosos *BAIXO* e *ALTO*. Foram empregadas as funções de mapeamento apresentadas nas figuras 3, 4 e 5.

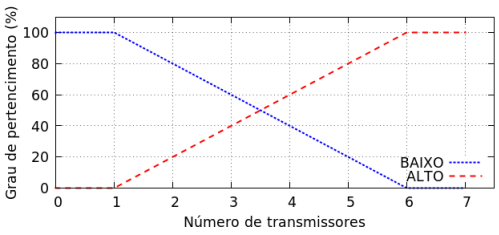


Figura 3. Funções de pertinência para número de transmissores.

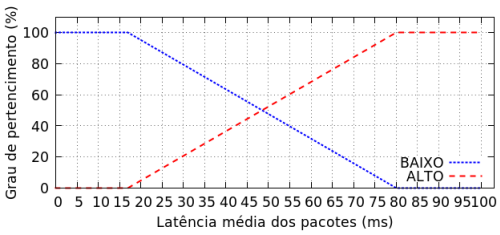


Figura 4. Funções de pertinência para latência média de entrega.

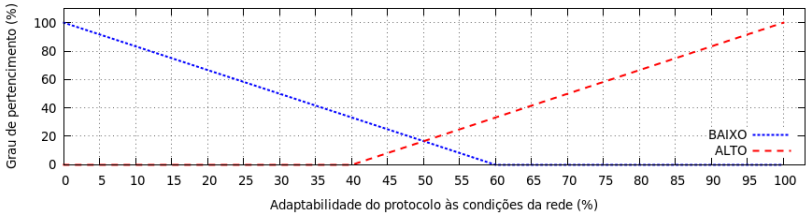


Figura 5. Funções de pertinência para adaptabilidade dos protocolos.

O motor de inferência, por sua vez, calcula o grau de adaptatividade de cada protocolo. As regras de inferência empregadas foram derivadas a partir da constatação de que técnicas de TDMA são adequadas quando temos muitos transmissores ou quando há um grande grau de contenção. A contenção é medida pela latência média, que captura a quantidade de retransmissões no meio. O motor de inferência utiliza as seguintes regras para calcular a adaptabilidade de cada protocolo:

```
CSMA
Se NT é BAIXO e LM é ALTO então ADP é ALTO
Se NT é ALTO e LM ALTO então ADP é BAIXO

TDMA
Se NT é BAIXO e LM é ALTO então ADP é BAIXO
Se NT é ALTO e LM é ALTO então ADP é ALTO
```

Esse conjunto de regras tem foco em redes onde a carga por estação é alta e possui pouca variação. Por consequência, essa implementação é efetiva em redes onde a carga total varia predominantemente em função da quantidade de transmissores.

A partir do cálculo da adaptabilidade de cada protocolo, um dicionário é construído e enviado ao módulo de Troca. Nessa estrutura, as chaves correspondem aos identificadores dos protocolos e os valores correspondem aos graus de adaptabilidade desses protocolos. A defuzzificação das regras é feita empregando o método da centroide.

4.4. Troca

O módulo de troca possui atribuições diferentes dependendo do tipo de nó (normal ou coordenador). O nó normal efetua a troca do protocolo localmente, enquanto o

nó coordenador, além dessa atribuição, coordena a troca do protocolo na rede e avisa periodicamente a todos os nós qual o protocolo está em atividade no momento.

Quando recebe o dicionário com as adaptabilidades vindo do módulo de Decisão, o módulo de Troca executa os passos descritos na seção 3.3. A troca só ocorre caso o grau do protocolo marcado como mais adaptado seja pelo menos 5% superior ao grau de adaptabilidade do protocolo atual. Isto é feito para evitar o efeito ping-pong, que ocorre quando o sistema oscila entre dois protocolos. Inicialmente o módulo de troca do nó coordenador envia uma mensagem do tipo *Envio de informações* ao protocolo ativo para que ele avise aos outros nós a necessidade de troca. Quando o módulo de troca recebe a mensagem do tipo *Informações enviadas*, ele inicia a troca local. A troca local envia uma mensagem do tipo *Parar atuação* ao protocolo ativo. O protocolo interrompe suas atividades e retorna a fila de envios juntamente com uma mensagem do tipo *Atuação parada*. O módulo de troca envia então uma mensagem do tipo *Iniciar atuação* juntamente com a fila de envios ao protocolo que deve entrar em operação. Esse protocolo inicia suas atividades utilizando a fila de envios recebida e responde com uma mensagem do tipo *Atuação iniciada* e a troca é concluída.

Após a troca, o coordenador informa em *broadcast* a cada 5 segundos qual protocolo está em atividade no momento. Essa mensagem periódica evita que algum nó permaneça muito tempo operando com um protocolo diferente.

5. Avaliação

Para avaliar a arquitetura FS-MAC, realizamos um experimento que mede a capacidade da plataforma de se adaptar a diferentes cenários de contenção na rede. Nesse experimento, a carga de dados de cada estação foi mantida constante e variou-se o número de estações transmitindo. Dessa forma, a carga total da rede, e consequentemente o seu grau de contenção, variaram principalmente em função do número de transmissores. Utilizamos sete estações, cada uma composta por um computador ligado a uma USRP Ettus B210. Os resultados estão expressos nos gráficos das figuras 8, 9, 10 e 11. Durante os testes, com até quatro estações transmitindo, a plataforma FS-MAC opera com o protocolo CSMA ativo, a partir de cinco transmissores ocorre a troca e o protocolo ativo passa a ser o TDMA.

A Figura 6 mostra o esquema de comunicação organizado durante o experimento. A seta indica a comunicação entre duas estações, a direção da seta indica a direção em que os dados estão sendo enviados e a estação E1 corresponde ao nó coordenador FS-MAC. A Figura 7 mostra como as URPS foram organizadas de acordo com o diagrama apresentado.

O experimento se inicia com um nó transmitindo dados e recebendo confirmação, na segunda etapa mais um nó é acrescentado à rede como transmissor. Nas etapas seguintes, outros nós são acrescentados um a um até um total de sete. Cada etapa dura 180 segundos, dos quais 60 são reservados para a estabilização do nó, instanciação de objetos, ativação dos blocos e formação da fila e durante os outros 120 segundo as métricas de avaliação são coletadas. Com o acréscimo dos nós aumenta-se a quantidade de tentativas de transmissão e consequentemente a contenção da rede. Esse processo foi realizado para os protocolos CSMA e TDMA separadamente e para a plataforma FS-MAC contendo os dois simultaneamente. Cada etapa foi repetida cinco vezes e foram coletados os valores

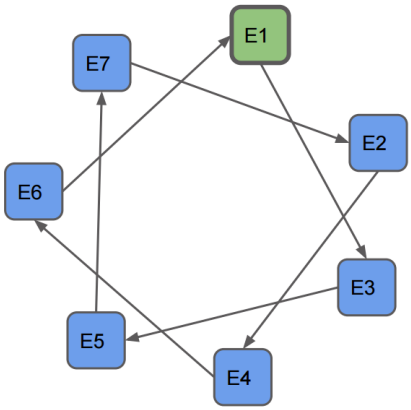


Figura 6. Diagrama de topologia e comunicação do experimento.

Figura 7. Organização das USRPs durante o experimento.

da quantidade de pacotes enviados, quantidade de pacotes confirmados, quantidade de retransmissões e a latência da entrega de cada pacote.

A Figura 8 apresenta a vazão dos protocolos avaliados. Os resultados observados mostram que os protocolos CSMA e TDMA separadamente possuem comportamento compatível com o esperado. Conforme observado em trabalhos como [Takagi and Kleinrock 1985], [Ziouva and Antonakopoulos 2002] e [Hu et al. 2011], o CSMA possui melhor desempenho quando a contenção é baixa, apresenta um ganho durante um intervalo de aumento de contenção, mas tem seu desempenho degradado à medida em que a contenção aumenta. Por outro lado, o TDMA possui baixo desempenho em cenários de baixa contenção, mas melhora à medida em que a contenção aumenta.

Na mesma figura podemos notar que o FS-MAC tende a seguir o protocolo que possui melhor desempenho em cada cenário. No início do experimento, onde a contenção é baixa, os desempenhos do CSMA e do FS-MAC são praticamente idênticos. Isso pode ser observado também nos gráficos das outras métricas avaliadas, como veremos a seguir. Analisando ainda o gráfico da Figura 8, a partir de três transmissores, o desempenho do FS-MAC é ligeiramente inferior ao desempenho do melhor protocolo. Notamos que com 3 e 4 transmissores a curva da plataforma FS-MAC passa abaixo do CSMA isolado, porém acima do TDMA. Nesses cenários, o FS-MAC está operando com CSMA. A partir do cenário com 5 nós transmitindo, a contenção da rede aumenta e assim o FS-MAC troca o protocolo para o TDMA.

Essa perda de desempenho em relação aos protocolos isolados deve-se principalmente às mensagens de controle do FS-MAC. As menagens de sensoriamento distribuído possuem maior impacto, pois embora não tenham grande relevância quando há poucos nós transmitindo, aumentam linearmente à medida em que os nós são inseridos na rede. Essa mensagem é enviada por cada nó normal a cada 10 segundos, gerando um grande *overhead* na rede. A mensagem de anúncio de protocolo ativo tem menor impacto, pois embora seja enviada a cada 5 segundos, parte apenas do nó coordenador, independente do tamanho da rede. Apesar do *overhead*, o desempenho do FS-MAC mantém uma distância relativamente pequena do melhor protocolo para cada cenário.

Como esperado, em relação às métricas de taxa de entrega e latência, o desempe-

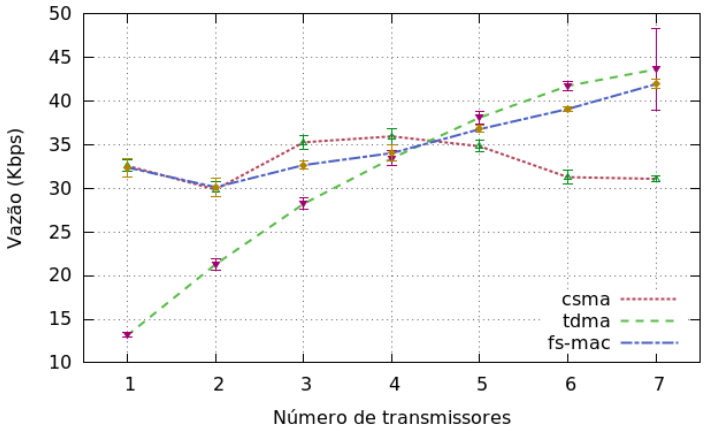


Figura 8. Vazão total da rede por número de transmissores.

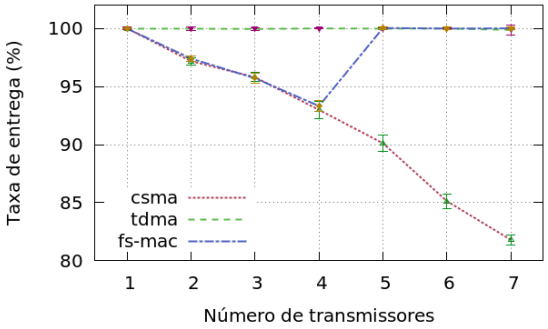


Figura 9. Taxa de entrega média dos pacotes por número de transmissores.

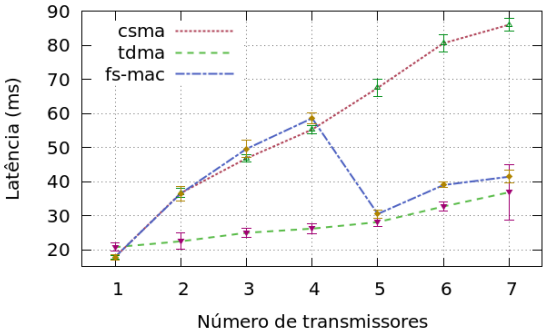


Figura 10. Latência média de entrega dos pacotes por número de transmissores.

nho do FS-MAC segue o do protocolo mais adaptado para o cenário. No caso da latência, mostrada na Figura 10, a perda está associada também ao envio de mensagens de controle. Essas mensagens podem interromper o reenvio de alguma mensagem que esteja na fila e não tenha sido entregue no primeiro envio.

Quanto à retransmissão de pacotes, ao contrário do esperado, o FS-MAC possui menor taxa que o CSMA no início do experimento. Nesse cenário, o número de envios de mensagem de controle do FS-MAC também é menor, assim é compreensível a menor taxa de retransmissão de pacotes mesmo tendo um desempenho ligeiramente inferior.

Em relação ao procedimento de troca dos protocolos, em todas as situações a plataforma detectou corretamente o protocolo a ser utilizado e a troca ocorreu em todos os nós, ou seja, nenhum nó permaneceu operando no protocolo antigo. Acreditamos que esse problema não ocorreu devido à topologia montada, onde todos os nós estavam a curtas distâncias entre si. Nos casos em que esse problema ocorre, o FS-MAC lida com ele fazendo o anúncio periódico de qual protocolo está em operação.

6. Conclusões e Trabalhos Futuros

Nesse trabalho apresentamos FS-MAC, uma arquitetura híbrida e com alta flexibilidade para a subcamada MAC de redes sem fio. Essa arquitetura é capaz de detectar,

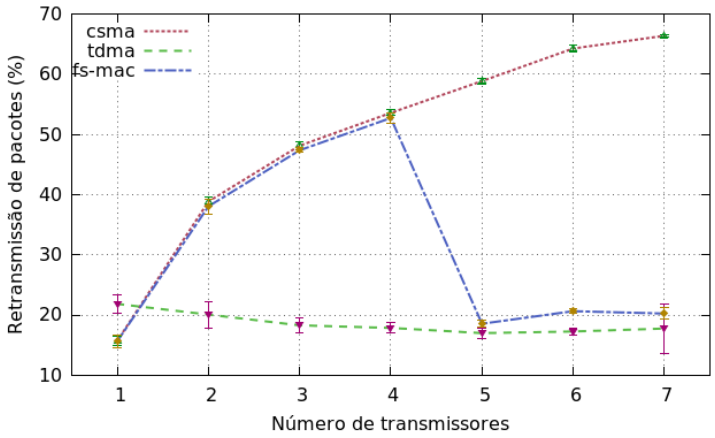


Figura 11. Taxa de retransmissão de pacotes por número de transmissores

através de um conjunto de regras, qual o protocolo mais eficiente para cada cenário apresentado na rede e colocar esse protocolo em atividade. A arquitetura é extensível, permitindo facilmente a adição de novos protocolos e também a alteração das regras de escolha, de acordo com as necessidades da aplicação e dos administradores da rede.

Para a avaliação da arquitetura implementamos um protótipo sobre o padrão IEEE 802.15.4, onde o FS-MAC escolhe entre um protocolo MAC baseado em contenção (CSMA) e outro livre de contenção (TDMA). Os resultados da avaliação mostram o correto funcionamento da arquitetura. Dadas as condições da rede, o protótipo é capaz de perceber qual é o protocolo mais eficiente e colocá-lo em operação corretamente. O comportamento dos protocolos dentro da arquitetura se mostrou muito similar ao seu funcionamento isolado. Devido ao *overhead* causado pelas mensagens de controle que garantem o seu funcionamento, FS-MAC apresenta um desempenho ligeiramente inferior em relação aos protocolos isolados nos cenários em que eles são mais eficientes.

Como trabalhos futuros, pretendemos analisar como incorporar o conceito de SDN no FS-MAC, de forma que um controlador defina regras para a troca de protocolos. Além disso, iremos analisar outros mecanismos para a tomada de decisão, como aprendizado de máquina, que permite que a arquitetura refine as suas escolhas com o tempo. Pretendemos também analisar cenários de redes com mais de um nó desempenhando o papel de coordenador FS-MAC e onde haja entradas e saídas dinâmicas de nós.

Referências

Diepstraten, W. and WCND-Utrecht, N. (1993). IEEE 802.11 wireless access method and physical specification. *Power*, 5:10.

Doerr, C., Neufeld, M., Fifield, J., Weingart, T., Sicker, D. C., and Grunwald, D. (2005). MultiMAC-an adaptive MAC framework for dynamic radio networking. In *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, pages 548–555.

Hu, W., Yousefi'zadeh, H., and Li, X. (2011). Load Adaptive MAC: A Hybrid MAC Protocol for MIMO SDR MANETs. *Wireless Communications, IEEE Transactions on*, 10(11):3924–3933.

Myers, A. D., Záruba, G. V., and Syrotiuk, V. R. (2002). An adaptive generalized transmission protocol for ad hoc networks. *Mob. Netw. Appl.*, 7(6):493–502.

Neufeld, M., Fifield, J., Doerr, C., Sheth, A., and Grunwald, D. (2005). SoftMac-flexible wireless research platform. In *Proc. HotNets-IV*, pages 1–5.

Nychis, G., Hottelier, T., Yang, Z., Seshan, S., and Steenkiste, P. (2009). Enabling MAC Protocol Implementations on Software-defined Radios. In *6th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 91–105.

Puschmann, A., Di Francesco, P., Kalil, M. A., DaSilva, L. A., and Mitschele-Thiel, A. (2013). Enhancing the Performance of Random Access MAC Protocols for Low-cost SDRs. In *8th ACM WiNTECH*, pages 9–16.

Rao, A. and Stoica, I. (2005). An overlay MAC layer for 802.11 networks. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 135–148.

Rhee, I., Warrier, A., Aia, M., Min, J., and Sichitiu, M. L. (2008). Z-MAC: A Hybrid MAC for Wireless Sensor Networks. *IEEE/ACM TON*, 16(3):511–524.

Ruckebusch, P., Giannoulis, S., Poorter, E. D., Moerman, I., Tinnirello, I., Garlisi, D., Gallo, P., Kaminski, N., DaSilva, L., Gawlowicz, P., Chwalisz, M., and Zubow, A. (2016). A unified radio control architecture for prototyping adaptive wireless protocols. In *European Conference on Networks and Communications (EuCNC)*, pages 58–63.

Schmid, T., Bloessl, B., and Wunsch, F. (2016). An IEEE 802.15.4 transceiver for GNU Radio v3.7. Disponível em: `https://github.com/bastibl/gr-ieee802-15-4`. Acesso em: 08/05/2016.

Sharp, B., Grindrod, E., and Camm, D. (1995). Hybrid TDMA/CSMA protocol for self managing packet radio networks. In *Fourth IEEE International Conference on Universal Personal Communications*, pages 929–933.

Silva, W. S., Cordeiro, J. R. S., Macedo, D. F., Vieira, M. A., Vieira, L. F., and Nogueira, J. M. S. (2016). Introdução a rádios definidos por software com aplicações em GNU radio. *Minicurso SBRC 2016*.

Takagi, H. and Kleinrock, L. (1985). Throughput analysis for persistent CSMA systems. *IEEE Transactions on Communications*, 33(7):627–638.

Tinnirello, I., Bianchi, G., Gallo, P., Garlisi, D., Giuliano, F., and Gringoli, F. (2012). Wireless MAC processors: Programming MAC protocols on commodity hardware. In *IEEE INFOCOM*, pages 1269 –1277.

Yang, Z., Zhang, J., Tan, K., Zhang, Q., and Zhang, Y. (2015). Enabling TDMA for today's wireless LANs. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1436–1444.

Ziouva, E. and Antonakopoulos, T. (2002). CSMA/CA performance under high traffic conditions: throughput and delay analysis. *Computer Communications*, 25(3):313 – 321.

Seleção Dinâmica de Interfaces baseado em Análise de Contexto para Redes Sem fio Heterôgeneas

Alex Monteiro[1], Eduardo Souto[1], Richard Pazzi[2]

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)

[2]Instituto de Tecnologia da Universidade de Ontario (UOIT) – Canadá

{alex.monteiro,esouto}@icomp.ufam.edu.br, richard.pazzi@uoit.ca

Abstract. *The Internet access has been increasingly common across different types of wireless technologies. However, this exploit multiple forms of access is not a simple task. Different parameters must be analyzed to deliver the best wireless network available to the end user. This paper proposes a new mechanism for selection of access networks, called CANS, which selects the best access interface based on user context analysis and device. The results show that the adoption of the proposed selection strategies enables the user to better exploit the advantages of wireless access technologies, achieving better ratesofflowandavailability ofaccess.*

Resumo. *O acesso à Internet tem sido cada vez mais comum através de diferentes tipos de tecnologias sem fio. Contudo, explorar esta multiplicidade de formas de acesso não é uma tarefa simples. Diferentes parâmetros devem ser analisados para entregar a melhor rede sem fio disponível ao usuário final. Este trabalho propõe um novo mecanismo para seleção de redes de acesso, denominado CANS, que seleciona a melhor interface de acesso baseado na análise de contexto do usuário e do dispositivo. Os resultados mostram que a adoção das estratégias de seleção propostas possibilitam ao usuário explorar melhor as vantagens das tecnologias de acesso sem fio, obtendo melhores taxas de vazão e de disponibilidade de acesso.*

1. Introdução

O acesso à Internet tem sido cada vez mais comum através de diferentes tipos de tecnologias de comunicação sem fio, sendo uma consequência natural do crescimento da cobertura de serviços e da evolução dos dispositivos móveis, atualmente equipados com múltiplas interfaces de acesso. Em geral, a satisfação do usuário está vinculada em acessar redes que ofereçam melhores velocidades de acesso, porém existem outras questões o consumo de energia e custo financeiro que impactam fortemente na satisfação deste. Em síntese, o que o usuário almeja é acessar a Internet com melhor velocidade de acesso, por mais tempo com menor consumo de energia e com menor custo financeiro.

A fim de atender as necessidades do usuário, diferentes estratégias têm sido propostas na literatura para selecionar a melhor interface de acesso. Estas propostas envolvem algoritmos de decisão de *handoff* vertical (troca de redes de acesso com tecnologias diferentes) que consideram múltiplos parâmetros em conjunto com RSS (*Recieved Strengh Signal*) tais como o SAW (*Simple Additive Weighting*) [Savitha and Chandrasekar 2011], TOPSIS [Tran and Boukhatem 2008] e o GRA (*Grey Relational*

Analysis). Esses métodos são simples e menos complexos de implementar, mas não são eficientes em aplicações de tempo real. Para tentar deixar as decisões mais eficientes e inteligentes, alguns trabalhos vêm sugerindo o emprego de estratégias de ciência do contexto para tentar sintetizar o conjunto de informações da rede e dos terminais como em [Kim et al. 2008][Saboji and Akki 2012][Li et al. 2013][Ward et al. 2014].

Devido a esta diversidade de fatores a serem analisados, ter a ciência do contexto em que o usuário está inserido é um ingrediente chave no processo de desenvolvimento de soluções que permitam que os dispositivos computacionais tomem as decisões adequadas e oportunas para os usuários [Kim et al. 2008] [Li et al. 2013]. O uso de informações contextuais é essencial na otimização do processo de gerenciamento de *handoff* vertical em ambientes formados por múltiplas redes sem fio. Tais sistemas devem se adaptar às mudanças e variações de contexto do usuário, tais como localização, capacidade do dispositivo, características das interfaces de acesso existentes nos dipositivos, níveis de conectividade de rede, requisitos da aplicação, entre outros.

Uma questão comum às soluções que lidam com múltiplas tecnologias de acesso sem fio existentes é que elas avaliam as diferentes redes de acesso como um todo, relacionando todas as interfaces em uma única metodologia de avaliação, sem efetivamente avaliar características intrísecas de cada tecnologia de acesso individualmente. Tal abordagem faz com que elas se tornem mais engessadas, dificultando a adesão de novas tecnologias de comunicação de acesso sem fio.

Neste contexto, este trabalho propõe um novo mecanismo para seleção de redes de acesso baseado na análise de contexto do usuário e do dispositivo. Este mecanismo, denominado CANS (*Context-Awareness Network Selection*), é capaz de selecionar a melhor interface de comunicação com base na análise de informações coletadas a partir da utilização do dispositivo, velocidade de deslocamento do usuário, consumo de banda e de energia. O CANS é composto por módulos de gerenciamento especializado para interface de rede. Esta modularidade torna o mecanismo escalável permitindo que novas estratégias de gerenciamento de interface possão ser adicionadas por meio da inclusão de novos módulos (interfaces de rede) quando necessário. As principais contribuições deste trabalho são:

1. Três novas estratégias de seleção e gerenciamento de interfaces: *Bluetooth*, Wi-Fi e acesso móvel 4G.
2. Um novo algoritmo de seleção de interface baseado na análise de contexto do usuário e do dispositivo.
3. Uma nova API para desenvolvimento de estratégias de *handoff* vertical e horizontal. A API CANS visa auxiliar desenvolvedores e pesquisadores a desenvolver novas soluções e melhoramentos no procedimento de *handoff*.
4. Uma aplicação que implementa e avalia as estratégias *de sele*ção e gerenciamento de interface no dispositivo do usuário, funcionando como uma extensão do gerenciador de rede do sistema operacional.

O restante deste artigo está organizado com segue: a Seção 2 apresenta alguns dos principais trabalhos relacionados à seleção de dinâmica de rede. A Seção 3 d*escreve* o CANS, suas características, estratégias de seleção e gerencimento das interfaces de acesso. A Seção 4 descreve a API desenvolvida e aplicação para prova de conceito da solução prosposta. A Seção 5 descreve os experimentos e resultados obtidos e a Seção 6 encerra este trabalho com as considerações finais.

2. Trabalhos Relacionados

Diferentes estratégias de decisão têm sido propostas para indicar quando uma rede ou interface candidata (ou talvez múltiplas redes, no caso de dispositivos *multihoming*) é mais adequada para o usuário. Tais estratégias podem ser categorizadas em [Akhila et al. 2012; Kassar et al. 2008]: estratégias baseadas na métrica RSSI, estratégias baseadas em função de custo, estratégias centradas no usuário, estratégias de decisão baseada em múltiplos atributos, lógica Fuzzy e redes neurais, e estratégia com ciência de contexto.

[Balbi *et al.* 2016] apresenta uma estratégia de seleção de rede baseada em RSSI a fim de reduzir o efeito ping-pong [Mhatre and Papagiannaki 2006] nas estações móveis. Na solução apresentada os autores modificam a ferramenta *WPA Supplicant* [Malinen 2013], responsável por realizar a autenticação, varreduras e *handoffs* entre pontos de acesso de Wi-Fi. A modificação ocorre na forma como *WPA Supplicant* realiza a escolha da rede de destino, uma estratégia de suavização das leituras de RSSI pela utilização da técnica de Média Móvel Exponencialmente Ponderada (MMEP) é utilizada. Tal solução apresentou significativa melhora com a redução do efeito ping-pong.

[Queiroz et al. 2015] propõe uma solução de *handoff* vertical utilizando o protocolo IEEE 802.21 *Media Independent Handover* (MHI) entre redes de móveis UMTS e WiMax. Nesta proposta o MHI é utilizado para gerenciar e reportar os eventos de rede, adotando uma estratégia de seleção de rede baseada em paramêtros de QoS. O objetivo é garantir que os requisitos de QoS para diferentes classes de serviços, como streaming de voz e vídeo, sejam atendidos.

Em alguns trabalhos, a estratégia de decisão é baseada em uma função de custo que mede o benefício obtido pelo *handoff* para uma rede em particular [Devi and Agrawal 2007]. Em geral, neste tipo de estratégia, a avaliação da rede candidata é realizada por uma soma de funções ponderadas de parâmetros específicos. Por outro lado, as estratégias centradas no usuário propõem critérios e políticas de decisão de *handoff* voltadas à satisfação do usuário [Tran and Boukhatem 2009][Awad et al. 2016].

Alguns desses critérios são adotados pelas estratégias onde a tomada de decisão é realizada por múltiplos atributos (MADM – *Multiple Attribute Decision Making*) [Tran and Boukhatem 2008]. A estratégia MADM é uma abordagem que trata a seleção dinâmica de interfaces com múltiplas alternativas (interfaces) e atributos (características das interfaces, preferências do usuário). Os mais populares métodos MADM são: o SAW [Savitha and Chandrasekar 2011], TOPSIS [Tran and Boukhatem 2008], AHP [Radhika and Reddy 2011]. Para tentar deixar as decisões mais eficientes e inteligentes, alguns trabalhos vêm sugerindo o emprego de estratégias de ciência do contexto para tentar sintetizar o conjunto de informações da rede e dos terminais, como em [Kim et al. 2008][Saboji and Akki 2012][Li et al. 2013][Ward et al. 2014].

Diferentemente das abordagens citadas, este trabalho propõe uma solução modular que utiliza estratégias auxiliares que selecionam a melhor rede acesso levando em conta a características específicas de cada tecnologia de acesso sem fio e depois seleciona a melhor interface de acesso baseado no contexto do usuário. Nossa meta é oferecer a melhor interface de acesso no que se refere à velocidade de acesso, consumo de energia e custo financeiro.

3. Seleção de Rede baseada em Ciência de Contexto.

O CANS (*Context-Awareness Network Selection*) é um mecanismo para gerenciamento de interface e seleção de redes de acesso em ambientes sem fio heterogêneos. O mecanismo proposto atua como um agente de software (um serviço) no dispositivo de comunicação móvel organizando informações do dispositivo (nível de bateria), do usuário (velocidade de deslocamento e utilização do dispositivo) e da rede (largura de banda). A partir da identificação dos cenários definidos pelo usuário, o CANS é capaz de estabelecer uma ordem preferencial de utilização das interfaces. Além do processo de seleção de interfaces, o CANS emprega estratégias de gerenciamento pontuais para as interfaces Bluetooth, Wi-Fi e de acesso móvel 4G/LTE.

3.1 Arquitetura

Na arquitetura do mecanismo proposto, ilustrada na Figura 1, o acesso e a gerência das interfaces de rede são realizados através das API's (*Application Programming Interface*) fornecidas pelo sistema operacional. O CANS trata o problema de seleção de redes em nível horizontal, quando ocorre a troca entre redes com mesma tecnologia de acesso (*handover* horizontal), e em nível vertical, quando ocorre a troca entre redes com diferentes tecnologias de acesso (*handover* vertical).

No nível vertical, a estratégia de seleção de interfaces é baseada em informações de contexto como velocidade de deslocamento do usuário, utilização do dispositivo pelo usuário a partir da identificação do estado do dispositivo (se o *display* está ligado/desligado), largura de banda em uso e nível de bateria do dispositivo. Tais informações são utilizadas para determinar a interface de acesso mais adequada ao usuário em um determinado momento.

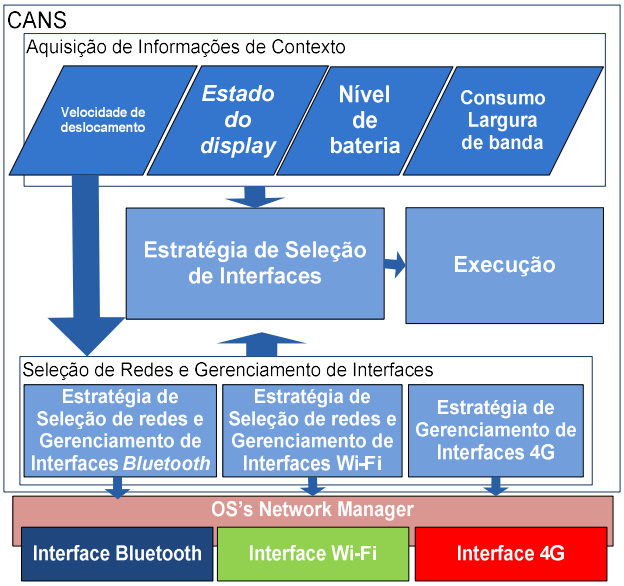


Figura 1. Arquitetura do CANS.

Por outro lado, no nível horizontal são aplicadas estratégias específicas para o gerenciamento da interface e seleção de rede para cada tecnologia de acesso, descritas na subseção 3.2. Tanto a estrátegia de seleção de interface, quanto as estratégias aplicadas invidualmente em cada interface possuem comunicação direta com o gerenciador de rede do sistema operacional do dispositivo.

Uma vez conhecida a melhor interface, o CANS executa a alteração do endereço IP do *gateway* e a interface padrão na tabela de roteamento do sistema operacional. As seções seguintes fornecem detalhes do funcionamento de cada estratégia utilizada pelo CANS.

3.2 Estratégias de Seleção e Gerenciamento de Interfaces

O CANS foi projetado para ser um mecanismo flexível e escalável, de modo que interfaces de acesso de outras tecnologias e novas estratégias de seleção e gerenciamento de interfaces possão facilmente ser adicionadas. Esta característica modular do CANS permite que cada estratégia trabalhe de forma pontual, de acordo com as particularidades de cada interface de acesso. O objetivo é entregar sempre à "Estratégia de Seleção de Interface" a interface já previamente conectada ao ponto de acesso e disponível (com acesso à Internet) para o processo de seleção.

3.2.1. Estratégia de Seleção e Gerenciamento para a interface Bluetooth

Esta estratégia tem a finalidade de identificar a disponibilidade de redes e dispositivos com tecnologia *Bluetooth* que permitam conectividade a serviços de rede e Internet com baixo consumo de energia. Para alcançar este objetivo, a estratégia identifica se o usuário encontra-se parado e inicia um processo de varredura do espectro a procura de dispositivos que possuam perfil de acesso NAP (*Network Area Personal*) e tenta realizar o pareamento automaticamente, como ilustrado na Figura 2. Este perfil permite que a interface possa ser utilizada como se fosse uma interface de rede de acesso Ethernet, permitindo a utilização do protocolo IP.

Figura 2. Etapas de funcionamento da estratégia de seleção e gerenciamento de interface Bluetooth.

Apesar de tecnologia Bluetooth disponibilizar diferentes classes de dispositivos que oferecem áreas de coberturas de aproximadamente 1 m (Classe 3), 10 m (Classe 2) e 100 m (Classe 3), a maioria dos dispositivos disponíveis no mercado são de classe 2, proporcionando uma área de cobertura limitada a 10 m. Considerando a limitação de área de cobertura, a estratégia adota que somente será vantajoso utilizar esta interface de acesso quando o usuário estiver estacionado, ou seja, sua velocidade de deslocamento for nula (igual a zero) durante um intervalo de tempo de *t* minutos.

3.2.2. Estratégia de Seleção e Gerenciamento para a interface Wi-Fi

A estratégia de gerenciamento de interface e seleção de redes Wi-Fi funciona de forma semelhante à estratégia para interface *Bluetooth*. Entretanto, em vez de entrar em

execução apenas quando o dispositivo está parado, o gerenciamento e a seleção dessa interface ocorre sempre que velocidade do deslocamento do usuário é menor ou igual a 5 Km/h ou 1,388 m/s, velocidade que corresponde a uma pessoa adulta caminhando [Mohler et al. 2007].

Conforme mostrado na Figura 3, para selecionar a rede Wi-Fi mais adequada, a estratégia realiza um sensoriamento do espectro coletando a força do sinal e o canal utilizado dos pontos de acesso adjacentes. Com base na força do sinal e dos canais utilizados, a estratégia calcula o fator de sobreposição de cada canal utilizado para descobrir a rede sem fio que possui o canal com menor interferência. Originalmente este fator de sobreposição é uma métrica comumente utilizada em vários algoritmos de seleção de canal para redes sem fio [Zhou and Liu 2012][Burton and Eng 2002] [Kalic et al. 2012][Monteiro et al. 2016]. Neste trabalho, o fator de sobreposição é utilizado sobre a pespectiva do nó cliente.

Uma vez que o fator de sobreposição é obtido, uma pontuação final da rede sem fio SR_n é estabelecida pela Equação 1,

$$S_{R_n} = \frac{Q_{Rn}}{Q_{max}} + \frac{RSSI - (-Sens)}{RSSI - Sens} + \frac{OF_{Max} - OF_{Rn}}{OF_{Max}} \quad (1)$$

onde Q_R_n é a qualidade da rede n, $RSSI$ é a força do sinal da rede n, OF_{max} é o fator de sobreposição máximo encontrado entre os canais, OFR_n é fator de sobreposição da rede n, $Sens$ é nível mínimo de sensibilidade da interface e Q_{max} é valor de qualidade máxima da interface dado pelo sistema operacional.

Após a atribuição da pontuação de todas as redes, caso o dispositivo não esteja conectado a nenhuma rede, a estratégia tenta estabelecer uma conexão com a rede com maior pontuação. Caso contrário, é realizada uma comparação entre a rede com maior pontuação com a rede em que o dispositivo está atualmente conectado. A estratégia se certifica que a troca de rede somente será realizada quando uma mesma rede for detectada com maior pontuação durante uma determinada quantidade de análises seguidas. O objetivo é evitar sucessivas trocas de redes ocasionando instabilidade ao mecanismo, também conhecido como efeito *ping-pong*. Por exemplo, para a implementação dessa estratégia neste trabalho foi adotado 4 (quatro) análises consecutivas com intervalo de 15 segundos, correspondendo ao tempo mínimo de um minuto para a troca da rede atual para uma rede com maior pontuação.

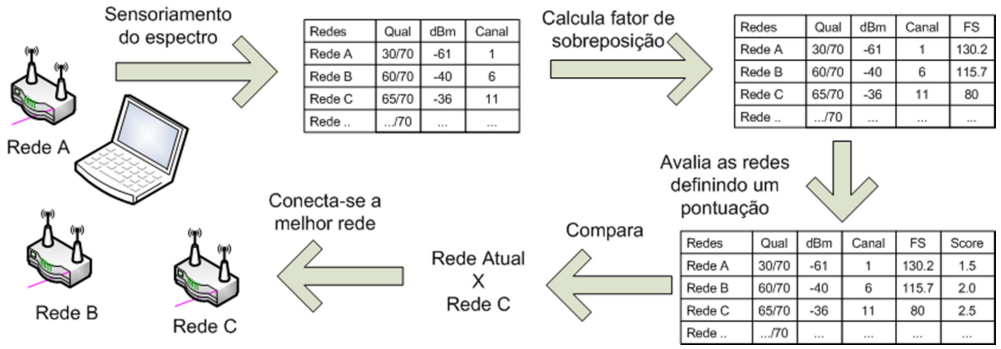


Figura 3. Etapas de funcionamento da Estratégia de Selecão de rede e Gerenciamento de interfaces Wi-Fi.

3.2.3. Estratégia de Gerenciamento para a interface de Acesso Móvel 4G

Diferente das estratégias apresentadas anteriormente, a estratégia para interface móvel (interface 4G) trabalha apenas gerenciando a ativação e a desativação da interface. Isto ocorre porque o *handover* horizontal da rede de acesso móvel é realizado somente pela operadora do serviço, cabendo ao CANS somente decidir quando tornar ou não a interface de comunicação disponível.

Por ser a interface que oferece a pior relação custo beneficio em relação à velocidade de acesso (vazão) e ao custo de acesso (MByte/R$), a interface de acesso móvel é selecionada como último recurso para acesso na maioria dos contextos definidos nesta proposta. A estratégia entra em ação sempre que as demais interfaces não estão ativas ou quando não possuem conexão com a Internet. Em compensação, quando o usuário está um deslocamento de alta velocidade (por exemplo, a partir de um automóvel) a rede de acesso móvel 4G proporciona maior estabilidade do que as redes locais e pessoais sem fio devido a sua área de cobertura mais abrangente. Para identificar a necessidade da utilização da interface, a estratégia verifica o estado de ativação e conexão de serviço das outras interfaces.

3.3. Estratégia de Seleção de Rede

Diferentes critérios podem ser levados em consideração para a etapa de decisão. Alguns desses critérios relacionados à rede, ao usuário, ao dispositivo e serviços em execução podem ser mais relevantes que outros dependendo do contexto. Antes da necessidade de realização de cálculos para quantificar as vantangens da utilização de determinada interface, a estratégia de seleção de rede verifica três parâmetros antes do processo de seleção, são eles: o estado da interface (ativo/inativo), o estado da conexão (conectado/desconectado ao ponto de acesso), e o estado do serviço (acesso à Internet disponível/indiponível).

O estado da interface permite identificar se a interface está ativada ou não para ser utilizada no processo de *handoff*. Caso não esteja, esta pode ser desconsiderada do processo de avaliação, ou seja, caso somente uma interface de acesso esteja ativada o processo de seleção nem chega a ser executado, a interface de acesso ativa já é selecionada imediatamente. O estado da conexão permite verificar se uma interface mesmo ativa possui uma conexão pré-estabelecida com algum ponto de acesso. Caso a interface não esteja vinculada a nenhum ponto de acesso, esta pode ser previamente descartada do processo de decisão. Por último, o estado do serviço permite identificar se algum serviço está disponível a uma determinada rede de acesso vinculada a uma determinada interface. Por exemplo, um dispositivo pode até estar conectado a um ponto de acesso (por exemplo, Wi-Fi), mas se o ponto de acesso não posssuir o acesso ao serviço desejado pelo usuário, por exemplo a Internet (requisito comum na maioria dos casos), a interface também é descartada do processo de decisão.

Após a checagem destes três principais estados e a confirmação que de que pelos menos duas interfaces estão aptas para o processo de seleção, a estratégia inicia a identificação do contexto e a escolha da interface adequada. A estratégia observa o contexto do dispositivo, a fim de garantir que o usuário se mantenha conectado o maior tempo possível com o menor custo financeiro e consumo de energia. Cada contexto é identificado a partir da combinação de quatro principais parâmetros: i) velocidade deslocamento; ii) utilização do dispositivo por parte o usuário; iii) consumo de banda total utilizado pelos serviços ativos; e iv) carga atual de bateria. A estratégia de seleção

de interface proposta define três políticas de contexto (velocidade de acesso, economia de energia e cobertura) que define a ordem de preferência de utilização das interfaces.

A Tabela 1 apresenta os valores e as condições necessárias para a aplicação de cada uma dessas políticas. Nessa tabela, B_T (*bandwith threshold*) é o valor referência que estabelece se o tráfego demandado pelo usuário é alto (>= B_T) ou baixo (< B_T), e P_B (*percentage battery*) é o valor referência que estabelece o nível crítico de bateria para aplicação da política de economia de energia.

Tabela 1. Políticas de contexto adotadas no CANs.

	Parâmetros				
Fonte de energia conectada	**Velocidade de Deslocamento Km/h**	**Em Uso**	**Largura de Banda**	**Nível de bateria**	**Política de Contexto**
Sim	< 5	-	-	-	Velocidade de Acesso
	>=5	-	-	-	Cobertura
Não	Parado 0	Sim	>=B_T	>=P_B	Velocidade de Acesso
				<P_B	Economia de Energia
		Não	<B_T	-	Economia de Energia
	Baixa velocidade 0 < v < 5	Sim	>=B_T	-	Velocidade de Acesso
			<B_T	>=P_B	Velocidade de Acesso
				<P_B	Cobertura
		Não	>=B_T	>=P_B	Velocidade de Acesso
				<P_B	Cobertura
			<B_T	-	Cobertura
	Alta velocidade > 5	-	-	-	Cobertura

A política "velocidade de acesso" visa abordar cenários em que o usuário está utilizando o dispositivo, se ele está parado ou em deslocamento com velocidade inferior a 5 Km/h. Esta política define a ordem de prioridade das interfaces baseado na velocidade de acesso, oferecendo ao usuário a rede de acesso com melhor velocidade. A política supõe que serviços de transferência de dados em tempo real ou não podem ser executadas em curtos espaços de tempo ou até mesmo simultaneamente. Com base nesta suposição, interfaces com maiores velocidades de transferência e redes que oferecem larguras de banda maiores são mais adequadas para estes contextos.

A política "economia de energia" visa maximizar a economia de energia quando o dispositivo não está sendo utilizado pelo usuário, mas necessita manter o dispositivo conectado a algum ponto de acesso devido à utilização de serviços de mensagens instantâneas e de comunicação. Com este entendimento esta política visa auxiliar o dispositivo na economia de energia, priorizando interfaces e redes de acesso com baixo consumo, mantendo preferencialmente uma única interface de acesso com menor consumo de energia disponível.

A política "cobertura" aborda o cenário em que a velocidade de deslocamento do usuário é superior a 5 Km/h. Esta política define a ordem de prioridade das interfaces baseada nas redes com maiores áreas de cobertura. Deste modo, excessivas execuções de *handoff* podem ser evitadas, o que pode maximizar a disponibilidade de serviço e minimizar as interrupções da comunicação.

A Tabela 2 apresenta a ordem de prioridade de seleção de interface, conforme as políticas de contexto apresentadas.

Tabela 2. Seleção das interfaces de acordo com as políticas de contextos.

Ordem de Preferência	Velocidade de Acesso	Cobertura	Economia de Energia
1ª	Wi-Fi	4G	Bluetooth
2ª	4G	Wi-Fi	Wi-Fi
3ª	*Bluetooth*	*Bluetooth*	4G

4. API e Protótipo CANS

O processo de *handoff* é um processo complexo tanto do ponto de vista de implementação como de execução, uma vez que vários parâmetros devem ser coletados e analisados até que a decisão sobre a interface e rede mais adequada possa ser executada. Desta forma, o objetivo da ferramenta desenvolvida é apoiar a aplicação e validação da metodologia proposta. A ideia é que o CANS possa ser usado como API pelos desenvolvedores de soluções *handoff*, por usuários finais por meio do protótipo apresentado a seguir, ou por pesquisadores como ferramenta de monitoramento do processo de troca de interfaces.

A arquitetura de implementação da API CANS, exibida na Figura 4, compreende em um conjunto de bibliotecas desenvolvidas para facilitar a coleta e análise de informações das interfaces através da disponibilização de estruturas e funções necessárias para o procedimento de *handoff*. Desta forma, um desenvolvedor pode utilizar a API CANS para desenvolver sua própria solução de *handoff* com base nos parâmetros utilizados neste trabalho.

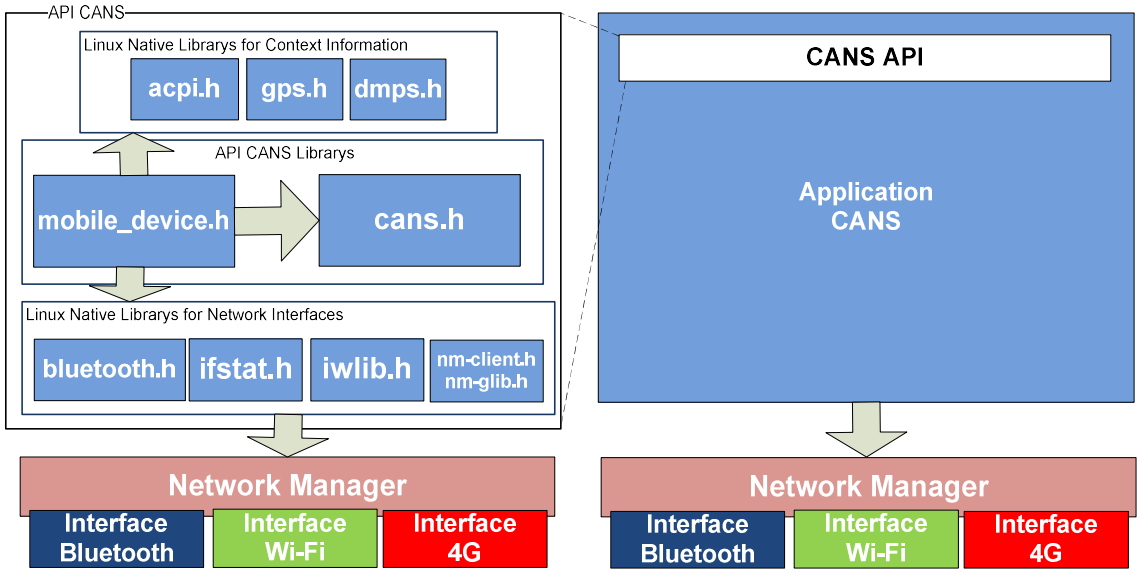


Figura 4. Componentes da API CANS.

Por exemplo, usando a API CANS, este trabalho desenvolveu uma aplicação que permite dois modos diferentes de operação: o modo monitoramento, que atua apenas como ferramenta de coleta de informações de contexto; e modo *handoff*, que coleta, avalia e executa o procedimento de *handoff* (horizontal ou vertical).

A aplicação, denominada AppCANS, atua como um sistema de suporte ao gerenciamento de rede do sistema operacional. A aplicação foi desenvolvida para plataforma Linux Ubuntu, onde seu atual sistema de gerenciamento de interfaces e de conexões da distribuição é *Network Manager*. Como constatado em outros gerenciadores de rede de outros sistemas operacionais, as funcionalidades contempladas pelo sistema é apenas o registro e o armazenamento de informações das conexões e redes uma vez já conectadas pelo usuário. Outro aspecto importante, é que todo o processo de ativação/desativação de interfaces e conexão/desconexão de novas redes de acesso não ocorre de forma automática, exigindo que haja interação por parte do usuário para acesso a novas redes de acesso. Com objetivo de proporcionar maior automação a este processo, a análise e seleção inteligente das interfaces de rede, a AppCANS faz uso das APIs "*libnm-glib*" e "*libnm-client*" que permite o gerenciamento das interfaces pelo *Network Manager* de forma automática com base nos resultados adotados pelas estratégias.

5. Experimentos e Resultados

O ambiente de experimentação real foi composto pela infraestrutura um dispositivo móvel (notebook Acer Aspire 5250 2,3 GHz, 4 GB) usando sistema operacional Linux Ubuntu 12.04, bateria com autonomia de 3560mAh e com as seguintes interfaces: i) *Bluetooth*: *Broadcom* BCM2046 2.1 EDR; 2) Wi-Fi: *Atheros Wireless* LAN 802.11n; e 3) 4G: Modem *Huawei* E1553.

A infraestrutura é composta pelas redes de acessos: *i*) Wi-Fi utilizando o roteador TPLink N600 a 300 Mbps, *ii*) *Bluetooth* utilizando o Agente Ecodroidlink para permitir a compartilhamento de acesso à Internet, instalados num Desktop com SO Ubuntu ; e *iii*) pela rede móvel 4G com velocidade de 1 Mbps.

5.1. Cenários de Teste

Os cenários de testes foram organizados de modo a observar e identificar os possíveis eventos que ocorrem no cotidiano dos usuários com seu dispositivo durante o deslocamento entre dois pontos.

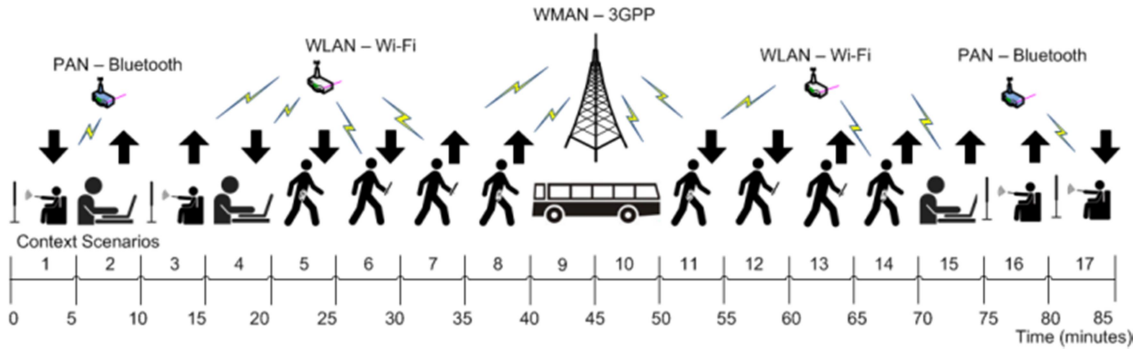


Figura 5. Cenários de Contexto utilizados nos experimentos.

O tempo de deslocamento total considerado foi de 1 hora e 25 minutos, dividido em 17 possíveis contextos com tempo de execução de 5 minutos, conforme exibido na Figura 5. Nesta figura, as setas direcionadas para baixo indicam baixo consumo de banda, ou seja, consumo inferior a um B_T = 1 Mbps, enquanto as setas direcionadas para cima indicam o alto consumo de banda, com valores superiores a B_T.

A Tabela 3 apresenta valores de velocidade, o estado de utilização do dispositivo e o consumo de banda para cada cenário de experimentação.

Tabela 3.Descrição dos cenários de contexto.

Contexto	Velocidade Km/h	Uso do dispotivo	Largura de banda Consumo Mbps
1 e 17	0	Não	< 1
2 e 15		Sim	>= 1
3 e 16		Não	>= 1
4		Sim	< 1
5 e 11	3,6	Não	< 1
6 e 12		Sim	< 1
7 e 13		Sim	>= 1
8 e 14		Não	>= 1
9	50	Sim	-
10		Não	

5.2. Resultados

A vazão proporcionada pelo mecanismo CANS é apresentada na Figura 6. Os resultados mostram que a estratégia de seleção de interfaces proposta no CANS permite ao usuário a constante disponibilidade de acesso. A variação das taxas de vazão ocorreu em função das mudanças de interface de acesso ocasionadas pela variação da velocidade de deslocamento do usuário. Durante o período de realização do *handoff* horizontal nas redes de acesso Wi-Fi e *handoff* vertical, executado entre as interfaces de acesso, as taxas de vazão foram nulas. Outra importante observação é que o CANS proporcionou uma disponibilidade de vazão semelhante à interface 4G, contudo oferecendo maiores velocidades de acesso, com picos de velocidade de transferência compreendidos entre 15 a 19 Mbps quando conectado a redes de acesso Wi-Fi.

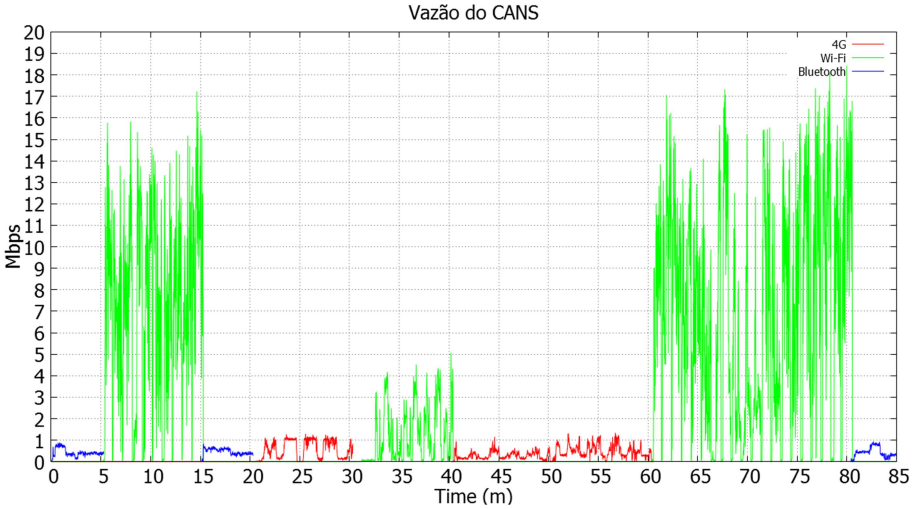


Figura 6 – Vazão de dados obtida pelo mecanismo CANS observando diferentes contextos.

A Figura 7 apresenta a quantidade de dados transmitidos e velocidade de acesso proporcionada por cada interface e pela utilização do CANS. A quantidade de dados transferidos pela utilização do CANS foi de 1,8 GBytes, enquanto que o total de dados transferidos pela interface *Bluetooth* foi de 150 MBytes e da interface 4G foi de 515 MBytes. A quantidade de dados transferidos pela interface Wi-Fi foi de 6,1 GBytes,

sendo 3,3 vezes superior a taxa de vazão obtida pelo CANS. Isto ocorre porque a estratégia de seleção do CANS procurar identificar a melhor interface de acesso considerando não apenas a vazão de dados, mas outros fatores como economia de energia e custo financeiro.

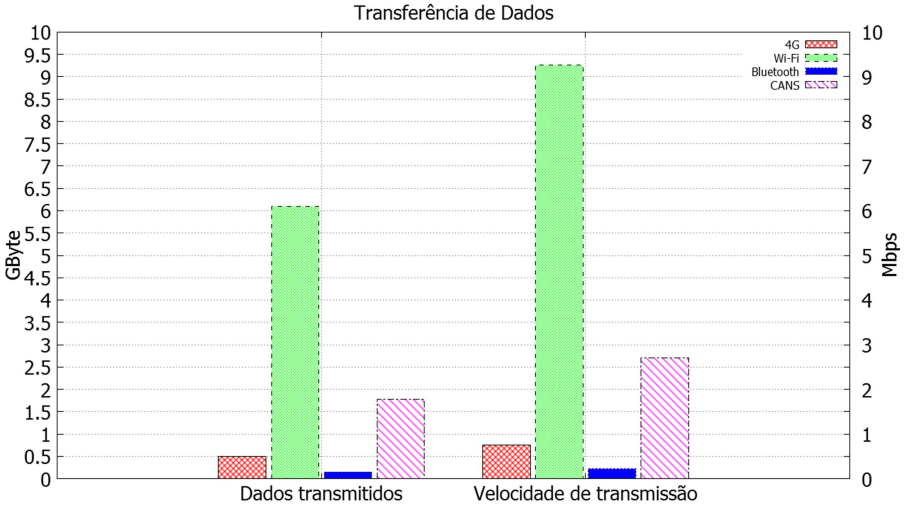


Figura 7. Comparação entre a Vazão obtida pela aplicação CANS e por outras interfaces de acesso.

A Figura 8 apresenta o percentual de disponibilidade de conexão por interface (sem utilizar o CANS) e utilizando o CANS. Este percentual é obtido consideram o tempo de conexão disponível divido pelo tempo total de cada experimento (1 h e 25 minutos). Os resultados mostram que a conectividade média proporcionada pelo CANS atinge até 95,13% de disponibilidade de conexão, sendo inferior somente a interface 4G com disponibilidade de 99,2 %. Este resultado ligeiramente inferior de 4% em relação a interface 4G corresponde justamente a indisponibilidade provocada pelo troca de interface durante o processo de *hard handoff*.

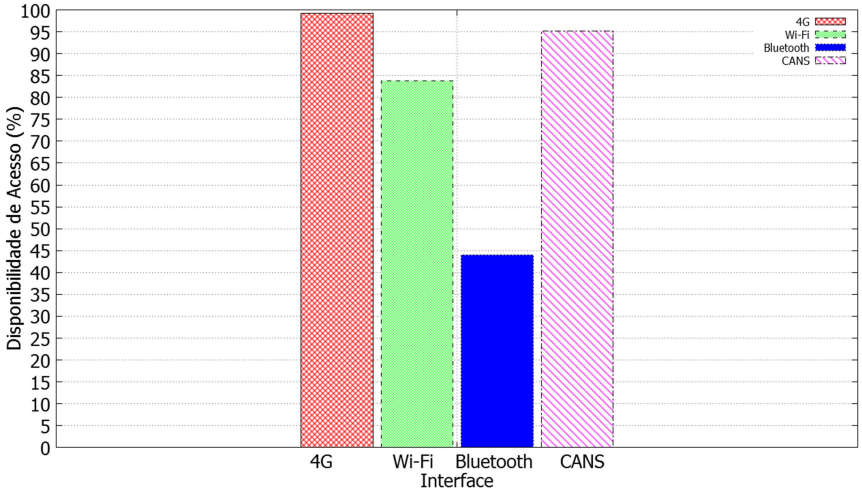


Figura 8. Disponibilidade de Conexão usando APPCANS.

6. Conclusões

Este trabalho apresentou o CANS, um mecanismo de seleção de interfaces baseado em análise de contexto a fim de proporcionar a rede e interface de acesso mais adequada ao

atual contexto do usuário. A partir da coleta informações de contexto do dispositivo, da rede e do usuário como velocidade de deslocamento do usuário, consumo de banda, nível de bateria do dispositivo e de utilização (estado do *display*), o CANS é capaz de aplicar estratégias de seleção e gerenciamento de interfaces para uma posterior execução de *handoff* vertical e horizontal.

Neste trabalho, a gerência de interfaces pode ser aplicada observando a velocidade de deslocamento do usuário. Dependendo das limitações da área de cobertura da tecnologia de comunicação empregada, uma interface pode ser ativada ou desativada de modo a otimizar a economia de energia do dispositivo e contribuir para a redução de determinados processos de relativos a etapa de decisão. Desta forma, os custos operacionais (processamento e tempo de execução) para os procedimentos de sensoriamento e de seleção de rede puderam ser reduzidos.

Os resultados demonstraram que a utilização das estratégias propostas possibilita o aumento de conectividade, oferecendo maior disponibilidade de serviço e maiores velocidades de acesso.

7. Agradecimentos

Este trabalho foi parcialmente financiado pela Comissão Européia e CNPQ através do projeto de pesquisa IMPRESS (FP7-ICT-2013-EU-Brazil, GA No. 614100), pela FAPEAM através do Projeto de Pesquisa PROTI Amazônia, e pelo Conselho de Pesquisa de Engenharia e Ciências Naturais do Canadá (NSERC) através do Programa NSERC Discovery Grant.

Referências

Akhila, S., Murthy, J., Shankar, A. e Kumar, S. (2012). An Overview on Decision Techniques for Vertical Handoffs across Wireless Heterogeneous Networks. *International Journal of Scientific & Engineering Research*, v. 3, n. 1, p. 1–6.

Awad, A., Mohamed, A. e Chiasserini, C. (2016). User-centric Network Selection in Multi-RAT Systems. *IEEE Wireless Communications and Networking Conference Workshops*, n. MEIoT, p. 97–102.

Balbi, H. Passos, D. Carrano, R., Magalhães, L. Albuquerque. V. N. (2016). Análise e solução para o problema da instabilidade de associação em redes IEEE 802.11 densas. *34rd Brazilian Symposium on Computer Networks and Distributed Systems*, p. 1–14.

Burton, M. e Eng, P. (2002). Channel Overlap Calculations for 802.11b Networks. *Cirond Technologies Inc.*,

Devi, M. K. S. e Agrawal, P. (2007). Dynamic Interface Selection in Portable Multi-Interface Terminals. *2007 IEEE International Conference on Portable Information Devices*, p. 1–5.

Kalic, G., Bojic, I. e Kusek, M. (2012). Energy Consumption in Android Phones when using Wireless Communication Technologies. *MIPRO, 2012 Proceedings of the 35th International Convention*, p. 754–759.

Kassar, M., Kervella, B. e Pujolle, G. (2008). An overview of vertical handover decision strategies in heterogeneous wireless networks. *Computer Communications*, v. 31, n. 10, p. 2607–2620.

Kim, B., Hong, J. e Cho, Y. (2008). Automatic Multi-Interface Management Through Profile Handling. *Mobile Networks and Applications*, v. 14, n. 1, p. 4–17.

Li, Q., Han, Q. e Sun, L. (2013). Context-aware handoff on smartphones. *Proceedings - IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems, MASS 2013*, p. 470–478.

Malinen, J. (2013). WPA Supplicant. https://w1.fi/wpa_supplicant/.

Mhatre, V. e Papagiannaki, K. (2006). Using Smart Triggers for Improved User Performance in 802.11Wireless Networks. *Proceedings of MobiSys*, p. 246–259.

Mohler, B. J., Thompson, W. B., Creem-Regehr, S. H., Pick, H. L. e Warren, W. H. (2007). Visual flow influences gait transition speed and preferred walking speed. *Exp. Brain. Res*, v. 181, n. 2, p. 221–228.

Monteiro, A., Souto, E., Pazzi, R. e Kiljander, J. (2016). Atribuição dinâmica de canais em redes sem fio não coordenadas IEEE 802 . 11 , baseada em fatores de sobreposição e intensidade de sinal. *XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, p. 704–717.

Queiroz, M., Atourassap, F., Reis, S., et al. (2015). Analysis of the integration of WiMAX and cellular networks. *LANOMS 2015 - 8th Latin American Network and Operations Management Symposium*, p. 77–83.

Radhika, K. e Reddy, A. (2011). AHP and Group Decision Making for Access Network Selection in Multi-Homed Mobile Terminals. *International Journal on Computer Science and Engineering*, v. 3, n. 10, p. 3412–3421.

Saboji, S. e Akki, C. (2012). Congestion-aware Proactive Vertical Handoff Decision Using Coalition Game. *International Journal of Soft Computing and Engineering*, n. 6, p. 91–97.

Savitha, K. e Chandrasekar, D. C. (2011). Vertical Handover decision schemes using SAW and WPM for Network selection in Heterogeneous Wireless Networks. *Global Journal of Computer Science and Technology*, v. 11, n. 9, p. 7.

Tran, P. N. e Boukhatem, N. (2008). Comparison of MADM Decision Algorithms for Interface Selection in Heterogeneous Wireless Networks. *16th International Conference on Software, Telecommunications and Computer Networks*, p. 25–27.

Tran, P. N. e Boukhatem, N. (2009). An utility-based interface selection scheme for multi-homed mobile terminals. *2009 IEEE 20th International Symposium on Personal, Indoor and Mobile Radio Communications*, p. 767–772.

Ward, P. a S., Naik, K. e Schmidtke, J. (2014). Efficient hashing for dynamic per-flow network-interface selection. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, p. 441–448.

Zhou, H. e Liu, C. (2012). WLAN Channel Assignment Based on Channel Overlap Factor. *2012 Second International Conference on Instrumentation, Measurement, Computer, Communication and Control*, p. 249–251.

Uma Abordagem Bidinâmica para a Identificação de Etiquetas RFID

Shalton Viana dos Santos, Paulo André da S. Gonçalves

Centro de Informática (CIn) – Universidade Federal de Pernambuco (UFPE)
50.740-560 – Recife – PE – Brasil

{svs2, pasg}@cin.ufpe.br

Abstract. *DFSA (Dynamic Framed Slotted ALOHA) is an access medium control protocol commonly used in RFID systems. Typically, the size of the frames under approaches without frame size resetting is dynamically adjusted in accordance with the specific value returned by the tag estimation method used. We propose a different and dual dynamic approach for the tag identification process. Our approch is doubly dynamic because it takes advantage of the dynamism of DFSA while using dynamic multiplicative factors for readjusting the specific value returned by the tag estimation method used. We evaluate our approach through simulation by using three different tag estimation methods. Time requirements of the communication channel are modelled in accordance with ISO 18000-6 Type C standard. The results show that our dual dynamic approach allows to optimize the system by minimizing the tag identification time under all the tag estimation methods evaluated.*

Resumo. *O DFSA (Dynamic Framed-Slotted Aloha) é um protocolo de acesso ao meio comumente usado em sistemas RFID. Tradicionalmente, o tamanho dos quadros é ajustado dinamicamente em função apenas do valor retornado pelo estimador de população de etiquetas empregado em abordagens sem reset de quadro. Este artigo propõe uma abordagem bidinâmica para o processo de identificação de etiquetas RFID. A proposta é duplamente dinâmica por contar com a dinamicidade no reajuste de tamanho de quadros do DFSA e por usar fatores multiplicativos dinâmicos no reajuste de estimativas retornadas pelo estimador utilizado no sistema RFID. As avaliações de desempenho da abordagem proposta foram realizadas através de simulação utilizando-se três estimadores distintos e requisitos temporais do canal de comunicação no padrão ISO 18000-6 Type C. Os resultados mostram que a abordagem bidinâmica permite otimizar o sistema para todos os estimadores estudados, reduzindo o tempo total de identificação de etiquetas.*

1. Introdução

A Internet das Coisas ou *Internet of Things* (IoT) é uma nova e promissora abordagem de rede composta por objetos físicos inteligentes (*e.g.* sensores, atuadores, eletrodomésticos, dispositivos eletrônicos, veículos, prédios, produtos em supermercados, produtos industrializados em geral) que podem, dependendo da aplicação, realizar comunicações autônomas, interagir entre si e trocar dados com a Internet [Perera et al. 2015] [Al-Fuqaha et al. 2015]. Existem aplicações de IoT no cotidiano onde é interessante que

exista alguma maneira de contar e identificar de forma rápida e automática uma determinada quantidade de objetos. Como exemplo, quanto mais rápido um caixa de supermercado contar e identificar os produtos de uma determinada compra, menores seriam as filas e mais satisfeitos ficariam os clientes com a eficiência dos caixas. Outro exemplo está após o processo de fabricação de grandes quantidades de produtos (*e.g.* canetas, parafusos, tampas), onde uma contagem automática deve ser rapidamente feita no momento de embalagem, distribuição e recepção pelo revendedor.

Os sistemas RFID (*Radio Frequency IDentification*) aparecem como os mais promissores no contexto de IoT não só para a identificação automática de objetos como também para outras aplicações. As principais razões para isso decorrem dos seguintes fatores: 1) a comunicação é através de sinais de radiofrequência (RF), não requerendo linha de visada direta; 2) o alcance de comunicação varia de centímetros a vários metros; 3) etiquetas RFID permitem suporte a aplicações de sensoriamento, localização, contabilização e indexação de objetos; 4) as etiquetas RFID são econômicas em termos de recursos de *hardware*, o que reduz custos financeiros; e 5) as etiquetas RFID podem funcionar sem bateria e sem necessidade de recarga constante caso usem bateria.

Os sistemas RFID mais simples são compostos por um leitor, possuindo ou não um servidor associado, e uma ou mais etiquetas. Os leitores podem ser ou não portáteis e, geralmente, não são tão restritos em termos de processamento e memória quanto as etiquetas. A depender do uso de bateria para a tarefa de comunicação, as etiquetas podem ser classificadas em passivas, semi-ativas ou ativas. As passivas não possuem bateria. Elas obtêm energia do sinal RF recebido do leitor e transmitem usando uma técnica conhecida por *backscatter*. As etiquetas ativas possuem bateria para comunicação e processamento. As etiquetas semi-ativas utilizam *backscatter* para comunicação e a bateria para processamento. Cada etiqueta possui um identificador (ID) único, podendo ser colada ou embutida em um objeto. E a depender da aplicação, o leitor requisita o ID das etiquetas em seu alcance de comunicação para identificação dos objetos. Nesse processo, como existe a possibilidade de haver colisões de transmissões, é necessário utilizar um protocolo anticolisão de etiquetas a fim de se resolver os conflitos de transmissão e permitir uma rápida identificação de todos os objetos.

Dentre os protocolos anticolisão para sistemas RFID, o DFSA (*Dynamic Framed Slotted ALOHA*) é um dos mais populares [EPC Global 2015, Klair et al. 2010]. A execução do DFSA é orientada pelo leitor que organiza o tempo em um ou mais quadros. Cada quadro é ainda subdividido em *slots* de tempo. As etiquetas são requisitadas a transmitir em um *slot*, aleatoriamente escolhido a cada quadro, até serem identificadas pelo leitor. Contudo, o DFSA requer o uso de alguma técnica para reajuste dinâmico do tamanho dos quadros. Existem vários estudos que fazem isso com base na estimativa da população de etiquetas competindo por *slots* [Andrade and Gonçalves 2013, Klair et al. 2010] ou com base em algum algoritmo [EPC Global 2015].

Os estimadores ou algoritmos de ajuste de tamanho de quadro para o DFSA em sistemas RFID podem ser divididos em duas classes básicas: com e sem *reset* de quadro. A primeira classe recalcula o valor do próximo quadro a ser aberto somente ao término da execução de um quadro completo [Andrade and Gonçalves 2013, Eom and Lee 2010, Li and Wang 2011, Wu and Zeng 2010, Tong et al. 2009, Chen 2009, Vogt 2002, Schoute 1983]. A segunda classe decide ao término de cada *slot* ou de um subgrupo de *slots*

consecutivos se cancela ou não a execução dos *slots* restantes e recalcula o tamanho do próximo quadro [Šolić et al. 2016, Chen 2014, EPC Global 2015].

Tradicionalmente, o tamanho dos quadros no DFSA na classe sem *reset* é ajustado dinamicamente em função apenas do valor retornado pelo estimador de população de etiquetas utilizado. Assim sendo, há apenas um fator dinâmico sendo utilizado no processo de identificação de etiquetas RFID. De forma diferente, este artigo propõe uma abordagem bidinâmica para esse processo. A proposta é duplamente dinâmica por contar com a dinamicidade no reajuste de tamanho de quadros do DFSA e por usar fatores multiplicativos dinâmicos no reajuste de estimativas retornadas pelo estimador utilizado no sistema RFID. Para demonstrar a potencialidade da proposta, foram realizadas avaliações de desempenho através de simulações com os estimadores clássicos *Lower Bound* e Schoute bem como com o estimador Eom-Lee, sendo este último um dos mais acurados já propostos na literatura. A modelagem do canal de comunicação no estudo de simulação é feita com base na estrutura de temporização do canal de comunicação do padrão ISO 18000-6 Type C, o qual é também conhecido como EPC Class-1 Generation 2 [EPC Global 2015].

O foco deste trabalho está em abordagens que analisam todo o quadro em execução para cálculo do tamanho do quadro subsequente. Os estudos apresentados neste artigo utilizam o *Lower Bound* e o Schoute devido ao fato de serem os estimadores mais simples já propostos e com custo computacional muito baixo. Em contrapartida, eles possuem baixa acurácia, o que reflete, geralmente, em uma maior quantidade de *slots* utilizados no processo de identificação de etiquetas em relação a estimadores mais acurados e pior tempo de identificação de etiquetas [Andrade and Gonçalves 2011]. Por outro lado, o estimador Eom-Lee possui um custo computacional elevado, sendo porém um dos estimadores mais acurados já propostos na literatura [Andrade and Gonçalves 2013, Eom and Lee 2010]. Desta forma, é possível mostrar o impacto da proposta deste artigo com tipos de estimadores distintos em relação ao custo computacional, à qualidade de estimação e ao tempo de identificação.

Os resultados alcançados mostram que a abordagem bidinâmica proposta permite otimizar o sistema com qualquer um dos estimadores sem *reset* estudados, reduzindo o tempo total de identificação de etiquetas. As principais contribuições deste artigo são a proposição de um otimizador de estimadores sem *reset* de quadros e a demonstração de que é possível se alcançar um desempenho similar ao do Eom-Lee utilizando o simples *Lower Bound* com o otimizador resultante da proposta bidinâmica e com custo computacional significativamente menor. O trabalho proposto é uma extensão e generalização de abordagem inicialmente proposta em [Andrade and Gonçalves 2011] que utiliza fatores multiplicativos fixos, independentes da saída do estimador, em vez de fatores multiplicativos dinâmicos, dependentes da saída do estimador do sistema RFID para reajuste de quadros. A otimização de abordagens derivadas do estimador Vogt [Andrade and Gonçalves 2013, Li and Wang 2011] ou que resetam o tamanho dos quadros em momento de execução a partir da análise de cada *slot* ou um subgrupo de *slots* consecutivos será abordada em trabalhos futuros.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados. A Seção 3 detalha e discute especificamente os estimadores *Lower Bound*, Schoute e Eom-Lee. A Seção 4 apresenta e discute a abordagem bidinâmica proposta para a identificação de etiquetas RFID. A Seção 5 apresenta as avaliações de

desempenho realizadas com canal de comunicação sob requisitos temporais do padrão ISO 18600-6 Type C. Por fim, a Seção 6 conclui este trabalho.

2. Trabalhos Relacionados

Esta seção apresenta os principais estimadores e algoritmos para o DFSA com e sem *reset* de quadros. Para facilitar a leitura, a Tabela 1 apresenta os principais parâmetros utilizados e o significado de cada um.

Tabela 1. Lista de Parâmetros.

Parâmetro	Significado
f	tamanho do quadro executado ou sendo executado
$\hat{f}$	tamanho do quadro subsequente calculado pelo estimador
$\hat{n}$	estimativa de etiquetas não identificadas (*backlog*) calculada pelo estimador
s_s	quantidade de *slots* bem sucedidos
s_v	quantidade de *slots* vazios
s_c	quantidade de *slots* em colisão

2.1. Sem Reset

A Figura 1 ilustra o funcionamento da classe sem *reset*. Basicamente, um quadro de tamanho f é aberto e cada etiqueta escolhe aleatoriamente um *slot* para transmitir informações. Ao término do quadro, caso haja ao menos um *slot* em colisão, executa-se um estimador para se determinar o tamanho $\hat{f}$ do quadro subsequente a ser utilizado. Os parâmetros utilizados como entrada para um estimador variam de acordo com a proposta. Contudo, essas entradas costumam ser os parâmetros s_v, s_s, s_c e f ou um subconjunto deles, onde s_v, s_s e s_c representam, respectivamente, o total de *slots* vazios, bem sucedidos e em colisão no quadro executado com tamanho f. O estimador calcula o tamanho $\hat{f}$ do quadro subsequente com base na estimativa do quantitativo de etiquetas que não foram identificadas no quadro de tamanho f executado. Essa estimativa é representada neste trabalho por $\hat{n}$.

Existem diversos estimadores sem *reset* de quadro propostos na literatura como o *Lower Bound* [Vogt 2002], o Schoute [Schoute 1983], o Vogt [Vogt 2002], o Eom-Lee [Eom and Lee 2010], o Chen [Chen 2009], o IV-II [Andrade and Gonçalves 2013], o CMEBE [Li and Wang 2011] e estimadores Bayesianos como os apresentados em [Wu and Zeng 2010] e [Tong et al. 2009]. Os estimadores *Lower Bound*, Schoute e Eom-Lee são utilizados nas avaliações de desempenho neste artigo e, por isso, são apresentados e discutidos separadamente na Seção 3. Os demais estimadores são apresentados a seguir.

O estimador Vogt [Vogt 2002] modela o processo de identificação de etiquetas com base em uma distribuição binomial. Basicamente, o estimador procura um quantitativo de etiquetas que minimiza a norma Euclidiana da diferença entre dois vetores: um com os valores de s_v, s_s e s_c e outro com os valores esperados para esses parâmetros. Esse quantitativo de etiquetas é a quantidade estimada de etiquetas que competiram por *slots* no quadro de tamanho f analisado. O tamanho $\hat{f}$ do quadro subsequente é igual a esse quantitativo menos s_s. Em [Eom and Lee 2010] é sugerido que o Vogt possui maior custo computacional do que o Eom-Lee. Contudo, a literatura ainda carece de estudos

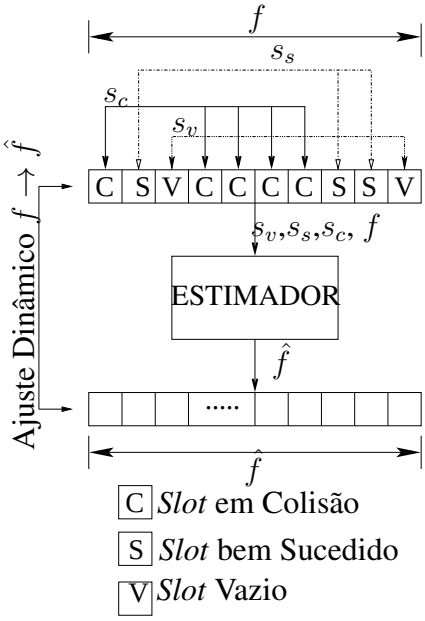


Figura 1. Abordagem sem *reset* de quadro.

mais detalhados com comparativos de custos entre o Vogt e o Eom-Lee em função do quantitativo de etiquetas a serem identificadas.

O IV-II [Andrade and Gonçalves 2013] identifica e trata a deficiência do estimador Vogt na produção de estimativa de etiquetas quando o quadro analisado possui todos os *slots* em colisão. Os resultados sugerem que o IV-II e o Eom-Lee são equivalentes em termos de quantidade de *slots* totais gerados no processo de identificação com um quadro inicial de 64 *slots*. Quando o quadro inicial é de 128 *slots*, os dois estimadores apresentam desempenho equivalente até 700 etiquetas. Acima disso, o IV-II apresenta melhor desempenho, chegando a usar até 69 *slots* a menos para identificar 1.000 etiquetas. Adicionalmente, o IV-II possui desempenho idêntico ou melhor do que o do Vogt, chegando a usar até menos 232 *slots* para identificar 1.000 etiquetas. O custo do IV-II é ligeiramente maior do que o do Vogt.

O estimador Chen [Chen 2009] também modela inicialmente o processo de identificação de etiquetas com base em uma distribuição binomial. Contudo, ele procura determinar a probabilidade de se obter exatamente s_v vazios, s_s sucessos e s_c colisões dado um quadro de tamanho f. A solução para esse problema deriva uma equação de probabilidades $P(n|f, s_v, s_c, s_s)$ que requer o cálculo de vários fatoriais e exponenciações. O valor de n que maximiza a probabilidade $P(n|f, s_v, s_c, s_s)$ é a estimativa de etiquetas que competiram por *slots* no quadro de tamanho f. Em geral, o estimador Chen usa mais *slots*, é menos acurado e mais custoso do que os estimadores Eom-Lee e IV-II [Eom and Lee 2010, Andrade and Gonçalves 2013]

Os estimadores em [Wu and Zeng 2010] e [Tong et al. 2009] se diferenciam dos demais por utilizarem um processo Bayesiano para a determinação da estimativa da quantidade de etiquetas que competiram por *slots* num quadro e para o cálculo do tamanho ótimo do quadro subsequente. A contrapartida é o custo computacional associado ao processo e, por isso, técnicas de redução de espaço de busca precisam ser utilizadas para

tornar viável o uso de estimadores desse tipo. Em [Wu and Zeng 2010] é mostrado como se reduzir o espaço de buscas e como se reduzir a complexidade computacional.

Em [Andrade and Gonçalves 2011] é apresentada a proposta de uma função de cálculo de tamanho de quadros para o DFSA. A função proposta reajusta o tamanho de quadro obtido pelo estimador empregado. Os estudos foram realizados com o *Lower Bound*, Schoute e Eom-Lee. A função ajusta o valor do tamanho de quadro da saída do estimador, multiplicando-o por um fator fixo cujo valor depende do estimador, mas independe do valor de saída do estimador. Os resultados mostram que é possível reduzir significativamente o tempo total de identificação de etiquetas dependendo da população de etiquetas a ser identificada e da relação temporal entre os diferentes tipos de *slots* de tempo.

Outros estimadores sem *reset* focam em melhorar a qualidade das estimativas em cenários com canais de comunicação sob efeito de captura como o CMEBE [Li and Wang 2011]. Esse estimador pode ser visto como uma variante do Vogt que considera um canal de comunicação com efeito de captura e calcula o tamanho $\hat{f}$ do quadro subsequente com base em uma estimativa $\hat{n}$ de etiquetas e uma estimativa da probabilidade média de captura em *slots* em colisão.

2.2. Com Reset

Outros estudos na literatura focam no desenvolvimento de estimadores com *reset*, onde o quadro de tamanho f, em execução, é analisado ao término de cada *slot* ou de um conjunto consecutivo de *slots*. Com base na análise realizada, o estimador pode decidir cancelar a abertura dos *slots* restantes e abrir imediatamente um novo quadro com um tamanho $\hat{f}$ recalculado. Comparada com a abordagem sem *reset*, a com *reset* incorre em processamento extra a cada *slot* ou conjunto de *slots* devido ao algoritmo de decisão de anulação de *slots*. Além disso, o cancelamento de *slots* aumenta o custo das mensagens do leitor para as etiquetas visto a necessidade de informá-las sobre o cancelamento de um quadro para a geração de novo número pseudoaleatório.

O estimador Chen II [Chen 2014] pode ser visto como uma versão do Schoute que analisa grupos consecutivos de *slots*. Ao analisar um grupo de *slots*, o estimador verifica se os parâmetros de desempenho obtidos a partir do grupo analisado correspondem ao esperado para se maximizar a eficiência do sistema. Caso não correspondam, os *slots* restantes são cancelados e um novo quadro é aberto. A estimativa $\hat{n}$ é igual a $(s_s + 2,39s_c)f/i$, onde $4 \leq i \leq f$ é o índice do último *slot* do subgrupo sendo analisado. O tamanho do quadro subsequente $\hat{f}$ é dado por $\hat{n}$ menos o quantitativo de *slots* bem sucedidos até o *slot* i. Caso não haja *reset* do quadro, o tamanho do quadro subsequente e a estimativa de etiquetas são obtidas conforme o estimador Schoute.

O padrão EPC Classe 1 Gen2 [EPC Global 2015] especifica dois procedimentos de identificação de etiquetas: um com quadros de tamanho fixo igual a 2^Q, ou seja, um FSA (*Framed Slotted Aloha*) simples e outro com quadros ajustados dinamicamente com tamanho 2^Q, porém, com Q recalculado ao término de cada *slot* com base no algoritmo-Q. Este segundo procedimento resolve ineficiências típicas de um sistema FSA e nada mais é do que um DFSA com *reset* de quadro ao término de cada *slot* vazio ou em colisão. Quando um *slot* termina, o leitor analisa o tipo de *slot*. De acordo com o resultado da análise, o leitor atualiza o valor do parâmetro Q_{fp} que nada mais é do que a versão em ponto flutuante

do parâmetro inteiro Q e é calculado como segue: $Q_{fp} = \min(15, Q_{fp} + C)$ para *slot* em colisão, $Q_{fp} = \max(0, Q_{fp} - C)$ para *slot* vazio e $Q_{fp} = Q_{fp} + 0$ para *slot* bem sucedido. O novo valor do parâmetro Q passa a ser $round(Q_{fp})$. Valores para C são dinâmicos e estão no intervalo de $0,1$ a $0,5$. A dificuldade está em se determinar valores adequados para C visto que valores ótimos são dependentes do número de etiquetas competindo por *slots*. O padrão não define como calcular valores adequados para C. Em [Khanna and Uysal 2015] é proposta uma modificação no algoritmo-Q. A proposta utiliza informações da camada física para estimar a quantidade de etiquetas que transmitiram em um mesmo *slot* e, assim, calcular um valor otimizado para o parâmetro Q.

O ILCM SbS [Šolić et al. 2016] é uma extensão de trabalhos anteriores dos autores e foca em produzir um valor Q otimizado para sistemas baseados no algoritmo-Q. Embora tenha como base uma equação de probabilidades $P(n|f, s_v, s_c, s_s)$ que também requer o cálculo de vários fatoriais e exponenciações, os autores conseguem encontrar funções menos custosas. Os resultados para um quantitativo de até 250 etiquetas mostram que o ILCM SbS permite um tempo de identificação inferior ao de uma versão do Vogt com *reset* de quadros.

3. Os Estimadores Estudados

Esta seção detalha e discute os estimadores *Lower Bound*, Schoute e Eom-Lee dado que serão os utilizados nas avaliações de desempenho neste artigo. Como explicitado na Seção 1, a escolha dos estimadores *Lower Bound* e Schoute é em virtude da simplicidade e do baixo custo computacional deles. Em contrapartida, são estimadores de baixa acurácia, o que reflete, geralmente, em um maior tempo de identificação de etiquetas em relação a estimadores mais acurados. Por outro lado, o estimador Eom-Lee possui um custo computacional significativamente maior, porém é um dos mais acurados já propostos na literatura. Assim, é possível avaliar o impacto da proposta neste artigo com estimadores distintos em relação ao custo computacional, à qualidade de estimação e ao tempo total de identificação de etiquetas. A Tabela 2 apresenta o custo de operações em ponto flutuante (FLOP) [Vales-Afonso et al. 2015] para análise dos estimadores.

Tabela 2. Custos de operações em ponto flutuante (FLOP).

Operação	Custo
adição, subtração, multiplicação	1
comparação	2
divisão, raiz quadrada	10
exponenciação, logaritmo	50
fatorial	100

3.1. Lower Bound

Formalmente, o estimador *Lower Bound* define [Vogt 2002]:

$$\hat{n} = s_s + 2 \cdot s_c \,, \tag{1}$$

$$\hat{f} = 2 \cdot s_c \,. \tag{2}$$

A lógica desse estimador é simples: há ao menos duas etiquetas envolvidas em uma colisão. Logo, a menor quantidade possível de etiquetas que competiram por *slots* em um quadro sendo analisado pode ser facilmente calculada, sendo igual a quantidade total de *slots* com transmissões bem sucedidas somada com o dobro da quantidade de *slots* em colisão. O *backlog* é a quantidade de etiquetas que ainda não foram identificadas e possui valor igual ao dobro da quantidade de *slots* em colisão. Para se maximizar a eficiência do sistema quando todos os *slots* possuem o mesmo tamanho, o quadro subsequente ao quadro sendo analisado deve ter tamanho igual ao *backlog* [Schoute 1983].

O *Lower Bound* é um estimador grosseiro, pois sempre estima a quantidade mínima possível de etiquetas restantes dentro de uma gama de possibilidades. Assim, ele pode ser visto como uma fronteira para se avaliar a qualidade de outros estimadores. Contudo, o custo FLOP dele é baixo. Com o auxílio da Tabela 2 é possível notar que o custo FLOP é igual 2 para o cálculo conjunto de cada tamanho de quadro e $\hat{n}$. Note que $\hat{n}$ é obtido através de simples soma tendo $\hat{f}$ já calculado. A Figura 2(a) apresenta o custo FLOP total de uso do estimador em função da quantidade de etiquetas a serem identificadas. Os resultados na Figura 2 são médias obtidas considerando-se a Tabela 2 e simulações de desempenho do estimador com os mesmos parâmetros apresentados na Seção 5, exceto pelo passo para o quantitativo de etiquetas. O intervalo de confiança é de 99% e é representado por barras de erro quase sempre imperceptíveis.

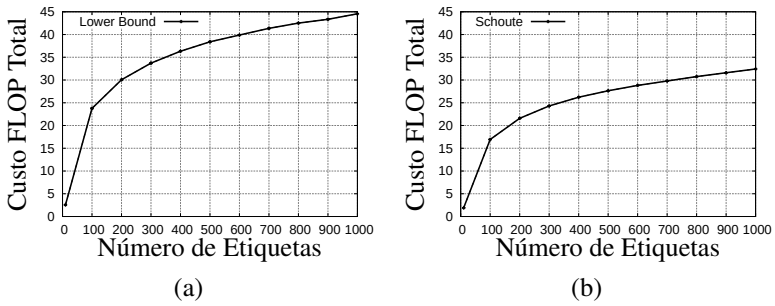


Figura 2. FLOP para os estimadores *Lower Bound* e Schoute.

3.2. Schoute

Formalmente, o estimador Schoute [Schoute 1983] define:

$$\hat{n} = s_s + 2,39 \cdot s_c \,, \tag{3}$$

$$\hat{f} = 2,39 \cdot s_c \,. \tag{4}$$

A mudança em relação ao *Lower Bound* está no uso do fator multiplicativo 2,39 ao invés de 2. Esse novo fator é o número esperado de etiquetas que transmitirão em cada *slot* em colisão no próximo quadro ao que está sendo analisado. A formulação de Schoute é obtida considerando-se um processo de chegadas do tipo Poisson. Para outras distribuições, as Equações (3) e (4) trazem resultados subótimos.

Note com base nas Figuras 2(a) e 2(b) que o Schoute possui, em geral, um custo FLOP total menor do que o do *Lower Bound* apesar de ambos terem o mesmo custo FLOP

no cálculo conjunto do tamanho do quadro e da estimativa de etiquetas. Isso ocorre porque o Schoute utiliza, em geral, menos quadros no processo de identificação de etiquetas. Assim, ele é executado menos vezes, impactando positivamente no custo FLOP total de uso do estimador. Para sistemas que requerem apenas o valor de $\hat{f}$, o custo FLOP total mostrado nas Figuras 2(a) e 2(b) cai pela metade.

3.3. O Estimador Eom-Lee

O estimador Eom-Lee [Eom and Lee 2010] utiliza um algoritmo iterativo tanto para estimar a quantidade de etiquetas que competiram por *slots* em um quadro analisado de tamanho L quanto para calcular o tamanho do quadro subsequente. O estimador Eom-Lee define que

$$\hat{f} = \gamma \cdot s_c \,, \tag{5}$$

onde

$$\gamma = \frac{1 - e^{-\frac{1}{\beta}}}{\beta(1 - (1 + \frac{1}{\beta})e^{-\frac{1}{\beta}})} \,. \tag{6}$$

Contudo, não é trivial encontrar uma solução fechada para se determinar os valores de γ e β a partir da Eq. (6). O problema é contornado calculando-se γ e β de forma iterativa. Considere γ_k e β_k, respectivamente, uma aproximação para o valor de γ e de β na k-ésima iteração do algoritmo. Essas aproximações são obtidas de acordo com as seguintes equações:

$$\beta_k = \frac{L}{\gamma_{k-1} \cdot s_c + s_s} \,, \tag{7}$$

$$\gamma_k = \frac{1 - e^{-\frac{1}{\beta_k}}}{\beta_k(1 - (1 + \frac{1}{\beta_k})e^{-\frac{1}{\beta_k}})} \,. \tag{8}$$

No primeiro passo do algoritmo iterativo, considera-se $\beta_1 = \infty$ e $\gamma_1 = 2$ e em cada passo k seguinte se determina uma nova aproximação para β e γ com o auxílio das Eqs. (7) e (8), respectivamente. Quando $|\gamma_{k^*-1} - \gamma_{k^*}|$ for menor que um limiar pré-definido $\epsilon_{threshold}$, o processo iterativo é interrompido. Os valores γ_{k^*-1} e γ_{k^*} representam, respectivamente, a aproximação anterior e atual para o valor de γ. A partir de então, o tamanho $\hat{f}$ do próximo quadro e a quantidade estimada $\hat{n}$ de etiquetas são obtidos, respectivamente, pelas Eqs. (9) e (10), onde β_{k^*} é a aproximação mais recente para o valor de β.

$$\hat{f} = \gamma_{k^*} \cdot s_c \,. \tag{9}$$

$$\hat{n} = \frac{\hat{f}}{\beta_{k^*}} \,. \tag{10}$$

O custo do Eom-Lee por cálculo de tamanho de quadro depende do número de iterações calculando-se β_k e γ_k até se atingir o critério de parada. Após atingir o critério de parada, há ainda o cálculo de $\hat{f}$ que adiciona 1 ao custo total das iterações e o cálculo de $\hat{n}$ que adiciona mais 10 ao custo. A Figura 3(a) apresenta o número total de iterações realizadas pelo estimador ao longo do processo de estimação de etiquetas. Note que o estimador possui um aumento abrupto de iterações a partir de 300 etiquetas. A Figura 3(b) apresenta o custo FLOP total de uso do estimador em função da quantidade de etiquetas a serem identificadas. Note que o custo FLOP total é significativamente maior do que o dos estimadores *Lower Bound* e Schoute. Para sistemas que requerem apenas o cálculo de $\hat{f}$, não há mudança significativa no custo visto que o número de iterações e cálculos nessas iterações possuem um peso muito mais importante. Os resultados na Figura 3 são médias obtidas considerando-se a Tabela 2 e simulações de desempenho do estimador com os mesmos parâmetros apresentados na Seção 5. Os intervalos de confiança são de 99%.

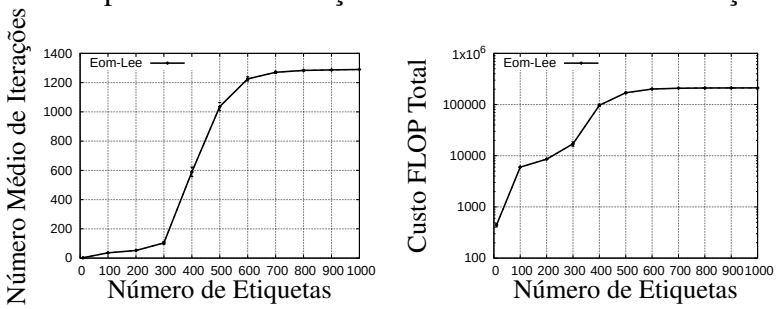


Figura 3. Número médio de iterações e custo FLOP total para o Eom-Lee.

4. A Abordagem Bidinâmica Proposta

Conforme mencionado na Seção 1, o tamanho dos quadros no DFSA é ajustado dinamicamente em função apenas de valor retornado pelo estimador de população de etiquetas utilizado. Desta forma, nota-se que há apenas um fator dinâmico sendo utilizado no processo de identificação de etiquetas. A proposta deste trabalho consiste em uma abordagem que adicionalmente analisa o valor retornado pelo estimador e o reajusta também dinamicamente tendo como base uma função de otimização aqui proposta. Assim, o bidinamismo da proposta é consequência do reajuste dinâmico de tamanho de quadros e do reajuste dinâmico da saída do estimador empregado. A Figura 4 apresenta uma esquematização da abordagem bidinâmica proposta.

Note que o diferencial em relação à Figura 1 está no módulo *Otimizador* que altera dinamicamente o valor de $\hat{f}$ que é retornado pelo estimador utilizado. O módulo *Otimizador* proposto neste artigo é uma generalização inspirada a partir dos achados apresentados em [Andrade and Gonçalves 2011]. Nesse trabalho, os autores demonstram ser possível reduzir o tempo total de identificação de etiquetas ao se multiplicar todo $\hat{f}$ retornado pelo estimador por um mesmo fator fixo calculado de forma adequada. Em vez de utilizar um mesmo fator fixo, o módulo *Otimizador* proposto usa um fator otimizado que varia de acordo com o resultado de saída do estimador.

Seja p a população de etiquetas a serem identificadas. Seja $\bar{\Omega}(i, p, \delta)$ uma função que representa o tempo médio total de identificação das p etiquetas utilizando-se um quadro inicial com i slots e um fator multiplicativo de correção de tamanho de quadro $\hat{f}$ igual

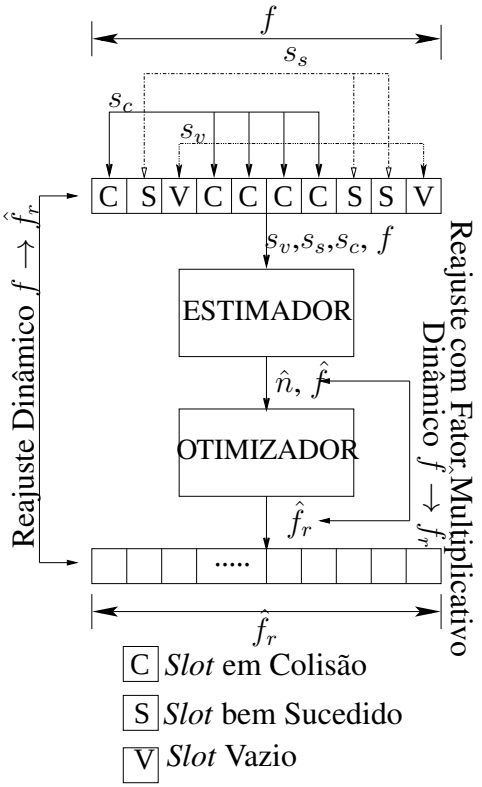


Figura 4. Abordagem bidinâmica para ajuste de tamanho de quadros.

a δ. A função $\bar{\Omega}$ é facilmente obtida através de simulação de Monte-Carlo para qualquer estimador desejado e seguindo o tamanho dos *slots* conforme norma utilizada pelo sistema RFID. A partir da função $\bar{\Omega}$, o tamanho do quadro corrigido dinamicamente para qualquer estimativa $\hat{n}$ e, respectivo tamanho de quadro $\hat{f}$, calculados pelo estimador empregado é dado por:

$$\hat{f}_r = \hat{f} \times \underset{\delta>0}{\operatorname{argmin}} \quad \bar{\Omega}(i, p, \delta) \quad \text{para} \quad p = \hat{n}, \tag{11}$$

significando que $\hat{f}_r$ é igual a estimativa $\hat{f}$ do estimador empregado multiplicada pelo valor de δ que minimiza o valor da função $\bar{\Omega}$ para $p = \hat{n}$, ou seja, para p igual ao valor da estimativa de etiquetas calculado pelo estimador.

A função $\bar{\Omega}$ não deve ser calculada em tempo de execução por causa de seu custo de processamento. Ela deve ser pré-computada e os dados produzidos podem ser pré-instalados em forma de uma árvore binária em memória com p sendo chave de busca para resultado pré-computado de $\operatorname{argmin}_{\delta>0} \quad \bar{\Omega}(i, p, \delta)$ e dado i. Isso permite que apenas simples consultas sejam feitas em tempo de execução, reduzindo-se de forma importante o tempo de processamento do otimizador. Portanto, o custo do otimizador proposto em relação ao uso exclusivo do estimador escolhido está no uso adicional de memória no leitor ou servidor associado, na realização adicional de uma multiplicação e na realização de uma busca em uma árvore binária na memória. Não há alterações nas etiquetas visto que o otimizador e o estimador são sempre executados no leitor RFID ou servidor associado. O custo de memória não é significativo. Cada nó da árvore precisa armazenar um único valor de δ para o valor de i definido no sistema RFID considerado. Valores típicos de i são 32,

64 e 128 *slots*. Além disso, o uso de árvore binária balanceada reduz significativamente o custo das consultas visto a complexidade de pior caso ser igual a $O(log\ p_{max})$, onde p_{max} é o número total de nós da árvore. Estudos na literatura consideram tipicamente a identificação de até 1.000 etiquetas, o que limita o valor de p_{max}.

5. Avaliação de Desempenho

Esta seção apresenta um estudo de simulação para avaliação do desempenho da abordagem bidinâmica proposta face a abordagem tradicional, considerando-se os estimadores *Lower Bound*, *Schoute* e Eom-Lee. Para isso, o simulador do protocolo DFSA para RFID desenvolvido por [Andrade and Gonçalves 2013] foi estendido com o módulo *Otimizador*. As simulações apresentadas consideram um sistema básico RFID com um leitor e um determinado quantitativo de etiquetas a serem identificadas. Esse quantitativo é variado de 10 a 1.000 em passos de 10. As métricas de avaliação são o tempo total de identificação e a redução média percentual no tempo total de identificação de etiquetas. Ambas as métricas consideram apenas o tempo gasto com *slots* vazios, em colisão e bem sucedidos. Esses tempos são os seguintes conforme norma ISO 18000-6 Type C: 1) tempo de *slot* bem sucedido = T_S = T4 +TQuery + 2T1 + 2T2 + TRN16 + TACK + TPC + EPC + CRC16 + TQREP = 2312 μs; 2) tempo de *slot* em colisão = T_C = T1 + T2 + TRN16 = 337,5 μs e; 3) tempo de *slot* vazio = T_E = T1 + T3 = 67,5 μs.

Nos gráficos apresentados, cada ponto em cada curva apresentada é uma média dos resultados de 2.000 simulações. O canal de comunicação é livre de erros visto o intuito de se estudar o impacto isolado dos estimadores e deles com o otimizador. Contudo, a proposta é aplicável com estimadores que levam em conta efeitos de captura, o que será melhor investigado em trabalhos futuros. Em particular às simulações com o estimador Eom-Lee, o parâmetro $\epsilon_{threshold}$ é igual a 0,001. Todas as simulações consideram um quadro inicial de 64 *slots*. Todos os resultados foram calculados com intervalo de confiança de 99%.

A Figura 5(a) mostra a redução média percentual no tempo total de identificação de etiquetas ao se utilizar a abordagem bidinâmica proposta. Cada curva revela o ganho obtido com a inclusão do otimizador em relação ao uso de cada estimador de forma isolada. Note que o otimizador consegue melhorar o tempo de identificação com os três estimadores e possui maior impacto com o *Lower Bound* visto que o ganho alcança perto de 9% para 1.000 etiquetas. Embora o Eom-Lee seja um estimador acurado, o gráfico mostra que alguma otimização ainda é possível. A Figura 5(b) apresenta o tempo total de identificação de etiquetas utilizando-se apenas as versões originais do *Lower Bound* e do Eom-Lee. O desempenho do Eom-Lee começa a se diferenciar de forma discreta do desempenho do *Lower Bound* a partir de 100 etiquetas. É importante observar que para diversos quantitativos de etiquetas, o maior custo FLOP total do Eom-Lee não implica em uma melhoria significativa no tempo total de identificação de etiquetas. Já para 300 e 1.000 etiquetas, por exemplo, o Eom-Lee permite um tempo total de identificação em torno de 6,3% e 8,5% menor, respectivamente. São ganhos consideráveis, mas a contrapartida é um custo de processamento significativo. As Figuras 5(c) e 5(d) apresentam um resultado interessante. Elas comparam o tempo total de identificação de etiquetas utilizando-se somente o Eom-Lee original com a versão do *Lower Bound* otimizada pela abordagem bidinâmica proposta. Note que o otimizador permite levar o *Lower Bound*, um estimador inacurado e de baixo custo computacional, a um desempenho similar ao do

Eom-Lee, um estimador acurado mas com custo computacional significativo. E isso trocando processamento por memória no leitor ou servidor associado e adicionando apenas uma multiplicação e uma busca em árvore binária balanceada que possui custo baixo.

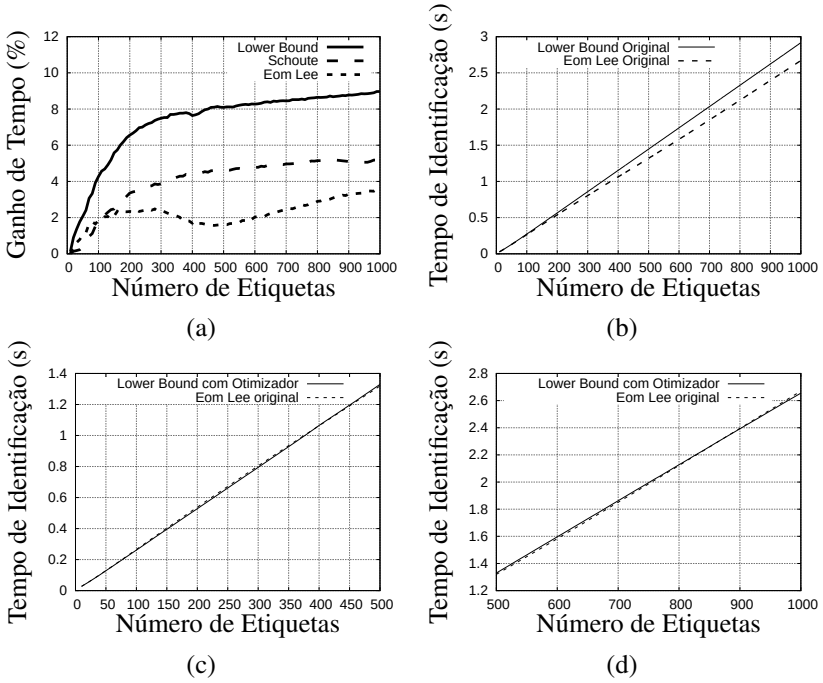


Figura 5. Resultados das Avaliações de Desempenho.

6. Conclusões

Este artigo propôs uma abordagem bidinâmica para a rápida identificação de etiquetas em sistemas RFID baseados no DFSA sem *reset* de quadros. A proposta otimiza dinamicamente a saída do estimador empregado pelo sistema RFID. As principais contribuições deste artigo foram a proposição de um otimizador de estimadores sem *reset* de quadros e a demonstração de que é possível se alcançar um desempenho similar ao do Eom-Lee utilizando apenas o simples *Lower Bound* associado ao otimizador resultante da proposta bidinâmica, mas com custo computacional significativamente menor do que o do Eom-Lee. Para isso, trocou-se processamento por memória e adicionou-se uma multiplicação e uma busca em árvore binária por tamanho de quadro calculado. Os trabalhos futuros incluem 1) o mapeamento do impacto da abordagem proposta com outros estimadores sem *reset*; 2) a extensão da proposta para uso com estimadores com *reset* e os impactos de desempenho e; 3) a extensão da proposta para sistemas baseados no algoritmo-Q e impactos de desempenho.

Referências

Al-Fuqaha, A., Guizani, M., Aledhari, M., and Ayysah, M. (2015). Internet of things: A Survey on Enabling Technologies, Protocols and Applications. *IEEE Communications Surveys Tutorials*, 17(4):2347 – 2376.

Andrade, J. D. and Gonçalves, P. A. S. (2011). Uma Função de Cálculo de Tamanho de Frames para o Protocolo DFSA em Sistemas RFID. In *Proc. of XVI Workshop de Gerência e Operação de Redes (WGRS)*, pages 61–74, Campo Grande, MS.

Andrade, J. D. and Gonçalves, P. A. S. (2013). Um Estimador Acurado para o Protocolo DFSA em Sistemas RFID. In *Proc. of XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 325–338, Brasília, DF.

Chen, W.-T. (2009). An Accurate Tag Estimate Method for Improving the Performance of an RFID Anticollision Algorithm Based on Dynamic Frame Length ALOHA. *IEEE Transactions on Automation Science and Engineering*, 6(1):9 –15.

Chen, W.-T. (2014). A Feasible and Easy-to-Implement Anticollision Algorithm for the EPCglobal UHF Class-1 Generation-2 RFID protocol. *IEEE Transactions on Automation Science and Engineering*, 11:485 – 491.

Eom, J.-B. and Lee, T.-J. (2010). Accurate Tag Estimation for Dynamic Framed-slotted ALOHA in RFID Systems. *IEEE Communications Letters*, 14:60–62.

EPC Global, I. (2015). *EPC radio-frequency identify protocols Generation-2 UHF RFID Specification for RFID air interface protocol for communications at 860MHz-960 MHz - ratified*, 2.0.1 edition.

Khanna, N. and Uysal, I. (2015). Q-frame-collision-counter: a novel and dynamic approach to RFID Gen 2's Q algorithm. In *Proc. of the IEEE International Conf. on RFID Technology and Application (RFID-TA)*, pages 120–125.

Klair, D., Chin, K.-W., and Raad, R. (2010). A Survey and Tutorial of RFID Anti-Collision Protocols. *IEEE Communications Surveys Tutorials*, 12(3):400 –421.

Li, B. and Wang, J. (2011). Efficient Anti-collision Algorithm Utilizing the Capture Effect for ISO 18000-6C Protocol. *IEEE Communications Letters*, 15:352– 354.

Perera, C., Liu, C. H., and Jayawardena, S. (2015). The Emerging Internet of Things Marketplace From an Industrial Perspective: A Survey. *IEEE Transactions on Emerging Topics in Computing*, 3(4):585 – 598.

Schoute, F. C. (1983). Dynamic Frame Length ALOHA. *IEEE Transactions on Communications*, 31:565–568.

Tong, Q., Zou, X., and Tong, H. (2009). Dynamic Framed Slotted ALOHA Algorithm Based on Bayesian Estimation in RFID System. In *Proceedings of the WRI World Congress on Computer Science and Information Engineering*, pages 384–388, Washington, DC, USA. IEEE Computer Society.

Vales-Afonso, J., Bueno-Delgado, V., Egea-Lopez, E., Gonzales-Castano, F., and Alcaraz, J. (2015). Multiframe maximum-likelihood tag Estimation for RFID Anticollision Protocols. *IEEE Transactions on Industrial Informatics*, 7(3):487 – 496.

Vogt, H. (2002). Efficient Object Identification with Passive RFID Tags. In *Proc. of the First International Conf. on Pervasive Computing*, pages 98–113, London, UK.

Šolić, P., Radić, J., and Rožić, N. (2016). Early Frame Break Policy for ALOHA-Based RFID Systems. *IEEE Trans. on Automation Science and Engineering*, 13(2):876–881.

Wu, H. and Zeng, Y. (2010). Bayesian tag estimate and optimal frame length for anti-collision aloha rfid systemas. *IEEE Transactions on Automation Science and Engineering*, 7(4):963 – 969.

Trilha Principal do SBRC 2017
Sessão Técnica 26
Tolerância a Falhas II

Um Protocolo Pessimista para Registro de Mensagens Baseado em um *Event Logger* Distribuído e Tolerante a Falhas

Edson Tavares de Camargo[1,2], **Fernando Pedone**[3], **Elias P. Duarte Jr.**[2]

[1]Universidade Tecnológica Federal do Parará - Campus Toledo (UTFPR)
CEP: 85902-490 – Toledo – PR – Brasil

[2]Universidade Federal do Paraná (UFPR) – Programa de Pós-Graduação em Informática
Caixa Postal 19081 – 81531-980 – Curitiba – PR – Brasil

[3]Università della Svizzera italiana (USI) – Faculty of Informatics
CH-6904 – Lugano – Switzerland

edson@utfpr.edu.br, fernando.pedone@usi.ch, elias@inf.ufpr.br

Abstract. *Most message logging protocols rely on a centralized event logger in order to store recovery information i.e, the determinants. This centralized approach, besides representing a single point of failure, is a bottleneck for the performance of the message logging protocol. In this work, we present a pessimistic message logging protocol built with a fault-tolerant distributed event logger based on consensus. Both are implemented and evaluated. Results show that the distributed event logger outperforms the centralized counterpart and that the proposed protocol allows efficient application recovery.*

Resumo. *A maioria dos protocolos de registro de mensagens conta com um event logger centralizado para armazenar informações de recuperação, isto é, os determinantes. Essa abordagem centralizada, além de apresentar um ponto único de falha, representa um gargalo para o desempenho dos protocolos de registro de mensagens. Neste trabalho, apresentamos um protocolo pessimista para registro de mensagens construído com um event logger distribuído e tolerante a falhas baseado em consenso. Ambos são implementados e avaliados. Resultados demonstram que o event logger distribuído tem desempenho superior ao da abordagem centralizada e que o protocolo proposto realiza a recuperação da aplicação eficientemente.*

1. Introdução

Rollback-recovery é uma técnica de tolerância a falhas tradicionalmente empregada em sistemas de alto desempenho e de longa duração [Egwutuoha et al. 2013]. O objetivo da técnica é restaurar o sistema a um estado consistente após uma falha [Elnozahy et al. 2002]. Para tanto, o estado de um processo é salvo periodicamente durante a sua execução e, perante uma falha, reiniciado a partir de um estado anterior, reduzindo assim a quantidade de trabalho perdido. O *checkpoint* é a ação de armazenar periodicamente o estado de um processo sem-falha em execução. Protocolos de registro de mensagens são uma categoria de *rollback-recovery* que, ao contrário da abordagem coordenada, não exigem a sincronização dos *checkpoints*. Além disso, na sua abordagem pessimista apenas o processo que falha é reiniciado. A abordagem garante uma

recuperação consistente a partir de *checkpoints* tomados independentemente em cada processo [Lifflander et al. 2014, Liu et al. 2013, Bouteiller et al. 2010].

Protocolos de registro de mensagens assumem que todos os eventos não-determinísticos que um processo executa podem ser identificados e a informação necessária para reaplicar cada evento durante a recuperação pode ser registrada em tuplas chamadas de *determinantes* [Elnozahy et al. 2002]. Considerando que os determinantes são salvos em uma entidade confiável, um processo em recuperação pode recriar deterministicamente o estado anterior à falha ao reaplicar os determinantes na sua ordem original. Consequentemente, uma tarefa crucial nos protocolos de registro de mensagens é salvar e recuperar de forma confiável os determinantes sem penalizar o desempenho da aplicação.

O componente responsável por armazenar de forma confiável os determinantes é chamado de *event logger*. O *event logger* recebe os determinantes dos processos da aplicação, armazena-os localmente, e notifica os processos da aplicação. Protocolos de registro de mensagens normalmente assumem que o *event logger* é uma entidade centralizada que não tolera falhas [Bouteiller et al. 2009, Ropars and Morin 2009, Bouteiller et al. 2005]. De fato, a falha de um *event logger* centralizado pode paralisar os processos da aplicação, uma vez que esses não mais conseguem salvar os determinantes.

O objetivo deste trabalho é apresentar um *event logger* distribuído e tolerante a falhas que tem desempenho comparável ou superior à abordagem centralizada, bem como um protocolo pessimista de registro de mensagens para interagir com o *event logger* proposto. Em particular, o *event logger* replicado não requer recursos extras (nodos físicos) em comparação a um *event logger* centralizado. Os *event loggers* replicados podem ser hospedados nos mesmos nodos dos processos da aplicação. Além disso, o *event logger* distribuído pode tolerar um número configurável de falhas.

Duas implementações do *event logger* proposto foram realizadas. Ambas são baseadas no algoritmo Paxos [Lamport 2001]. A primeira implementação se apoia em uma configuração tradicional do Paxos, chamadas de Paxos clássico. A segunda configuração é chamada de Paxos paralelo. As duas implementações são comparadas com uma implementação de um *event logger* centralizado. Um protocolo pessimista de registro de mensagens é implementado em MPI para interagir com o *event logger* proposto. O MPI é o padrão de *facto* para o desenvolvimento de aplicações paralelas e distribuídas de alto desempenho [Fagg and Dongarra 2000]. Entre as principais características do protocolo proposto estão as seguintes: a distinção entre eventos determinísticos e não-determinísticos em MPI; o emprego da abordagem de *checkpoint* não-coordenado; e a recuperação automática da aplicação perante falhas de processos. Os *event loggers* foram avaliados usando as aplicações LU e MG do *NAS Parallel Benchmark*, a aplicação AMG (*Algebraic MultiGrid*) e o algoritmo paralelo de Gusfield. O algoritmo de Gusfield também foi empregado para avaliar a recuperação. Resultados demonstram que o *event logger* baseado na abordagem Paxos Paralelo tem desempenho superior ao *event logger* centralizado e que o protocolo proposto realiza a recuperação da aplicação eficientemente.

Este trabalho segue organizado da seguinte forma. A Seção 2 apresenta os conceitos principais da técnica de *rollback-recovery*. A Seção 3 detalha o protocolo proposto, incluindo o *event logger* baseado em consenso. A Seção 4 aborda a implementação e a Seção 5 os resultados experimentais. Por fim, a Seção 6 apresenta a conclusão.

2. A Técnica de *Rollback-Recovery*

A técnica de tolerância a falhas *rollback-recovery* é frequentemente empregada para prover tolerância a falhas em aplicações de alto desempenho. Desse modo, as aplicações podem reiniciar a partir de um estado salvo previamente. A técnica assume um sistema distribuído, onde os processos da aplicação se comunicam através de uma rede e têm acesso a um dispositivo de armazenamento confiável que sobrevive a falhas [Elnozahy et al. 2002]. Periodicamente, os processos salvam informações de recuperação no dispositivo confiável durante a sua execução sem-falhas. Após a ocorrência de uma falha, a aplicação usa as informações de recuperação para reiniciar a sua computação a partir de um estado anterior. As informações de recuperação incluem os *checkpoints*, isto é, os estados dos processos participantes. Alguns protocolos também incluem o registro de eventos não-determinísticos, codificados em tuplas chamadas de *determinantes*. Basicamente, um estado global consistente é aquele em que se o estado de um processo reflete uma mensagem recebida, então o estado correspondente do emissor deve refletir o envio daquela mensagem [Kshemkalyani and Singhal 2011]. Um conjunto de *checkpoints* que corresponde a um estado consistente é chamado de *linha de recuperação*. O principal objetivo dos protocolos de *rollback-recovery* é restaurar o sistema a partir da mais recente linha de recuperação após uma falha. Os protocolos de *rollback-recovery* podem ser classificados em duas categorias: baseados somente no *checkpoint* (*checkpoint-based*) ou baseados em registro de mensagens (*log-based*), descritos a seguir.

2.1. *Rollback-Recovery* Baseado em *Checkpoints*

Os protocolos de *rollback-recovery* baseados em *checkpoints* dividem-se em duas abordagens: coordenada e não coordenada. Quando o *checkpoint* é realizado independentemente em cada processo, sem uma coordenação global, é chamado de não coordenado. A vantagem da abordagem não coordenada está em cada processo criar o seu *checkpoint* quando lhe é mais conveniente. Entretanto, com essa abordagem, um estado global consistente pode nunca ser atingido. Nesse caso, os *checkpoints* realizados tornam-se inúteis e devem ser descartados [Elnozahy et al. 2002, Kshemkalyani and Singhal 2011].

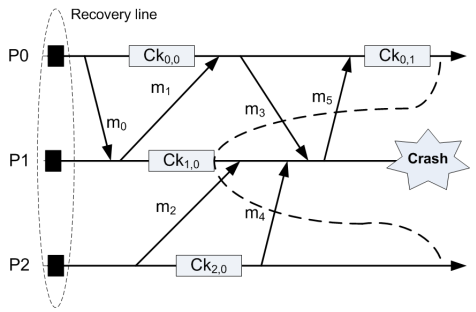


Figura 1. *Checkpoint* não coordenado e um linha de recuperação.

A Figura 1 apresenta um cenário no qual o mais recente conjunto de *checkpoints* (isto é, $Ck_{0,1}$, $Ck_{1,0}$, e $Ck_{2,0}$) não resulta em uma linha de recuperação. Isso se deve ao fato de que a mensagem m_5 é recebida por P_0 mas não enviada por P_1 — nesse caso, m_5 é chamada de uma *mensagem órfã* e P_0 um *processo órfão*. P_0 é então obrigado a retroceder a um *checkpoint* anterior (isto é, $Ck_{0,0}$). O problema entretanto persiste, pois

m_1 e m_2 precisam ser retransmitidas. Desta forma, a única linha de recuperação corresponde ao estado inicial da aplicação. Esse fenômeno é conhecido como *efeito dominó*. O *checkpointing* não coordenado é suscetível ao efeito dominó.

O *checkpointing* coordenado evita o efeito dominó. Os processos se sincronizam para realizar os *checkpoints* e, consequentemente, criar um estado global consistente [Kshemkalyani and Singhal 2011]. Embora a abordagem coordenada seja relativamente fácil de implementar, a sua execução impõe uma sobrecarga considerável à aplicação uma vez que os processos precisam se coordenar e salvar os seus estados simultaneamente no dispositivo de armazenamento. Além disso, mesmo que um único processo falhe, todos os processos precisam retroceder ao último *checkpoint*.

2.2. *Rollback-Recovery* Baseado em Registro de Mensagens

Os protocolos de *rollback-recovery* baseados em registro de mensagens empregam tanto *checkpoints* quanto o registro de eventos não-determinísticos com o objetivo de evitar as desvantagens das abordagens coordenada e não coordenada. Os protocolos de registro de mensagens assumem que todos os eventos não-determinísticos executados por um processo podem ser identificados e a informação necessária para reproduzir cada evento durante a recuperação pode ser registrada em determinantes [Kshemkalyani and Singhal 2011]. Um *evento* corresponde a um passo de comunicação ou um passo de computação de um processo. A maioria dos protocolos de registro de mensagens assume que a recepção das mensagens é o único evento não-determinístico. O registro de mensagens evita o efeito dominó do *checkpointing* não coordenado salvando todas as mensagens recebidas. Por exemplo, na Figura 1, as mensagens m_2, m_4 e m_3 recebidas pelo processo P_1 devem ser salvas, assim como os determinantes que contém a ordem de recepção das mensagens. Durante a recuperação somente o processo P_1 retrocede. Assim, o estado de P_1 eventualmente será o mesmo ao anterior à falha uma vez que as mensagens m_2, m_4 e m_3 são reaplicadas na mesma ordem.

Dependendo de como os determinantes são registrados, os protocolos de registro de mensagem podem ser classificados em pessimista, otimista ou causal [Elnozahy et al. 2002]. Na abordagem pessimista do registro de mensagens um processo primeiro armazena o determinante antes de entregar a mensagem. Apesar de simplificar a recuperação e a coleta de lixo, a abordagem pessimista gera uma sobrecarga durante a execução da aplicação: a aplicação precisa aguardar pelo armazenamento de cada determinante.

Como descrito em [Bouteiller et al. 2010], é possível reduzir o número total de determinantes armazenados distinguindo os eventos determinísticos dos não-determinísticos. Um evento é determinístico quando a partir do estado atual existe somente um estado resultante possível para o evento. Se um evento pode resultar em estados diferentes, então é dito não-determinístico. A recepção de mensagens com uma identificação de emissor explícita é um evento-determinístico e não requer o seu registro. Ao contrário, quando se aguarda uma mensagem de um emissor desconhecido então a recepção é dita não-determinística. Conforme definido em [Cappello et al. 2010], muitas aplicações MPI contêm somente eventos de comunicação determinísticos. Porém, muitas importantes aplicações MPI são não-determinísticas. Além disso, os desenvolvedores geralmente incluem o não-determinismo na codificação para melhorar o desempenho da aplicação.

3. Um Protocolo para Registro de Mensagens Baseado em Consenso

O *event logger* desempenha um papel crucial nos protocolos de registro de mensagens [Bouteiller et al. 2005]. O *event logger* recebe os determinantes, armazena-os localmente, e notifica os processos da aplicação após armazená-los. Apesar do *event logger* exercer este grande impacto, muitos protocolos o implementam como um componente centralizado e incapaz de tolerar falhas [Ropars and Morin 2009, Bouteiller et al. 2006, Ropars and Morin 2010]. Uma vez que *event logger* precisa notificar os processos da aplicação ao salvar um determinante, um *event logger* centralizado facilmente se torna em um gargalo conforme aumenta o número de processos. Além disso, a falha do *event logger* pode paralisar a aplicação ou levá-la a um estado inconsistente durante a recuperação. Nesta seção apresentamos o modelo do sistema, a descrição do protocolo proposto e do serviço de *event logger* propriamente dito.

3.1. Modelo de Sistema

Considere um sistema representado por um grafo unidirecional completo $G = (V, E)$, o conjunto de vértices V corresponde ao conjunto de processos e uma aresta $(i, j) \in E \mid i, j \in V$ representa a habilidade dos processos i e j de se comunicar diretamente, sem intermediários. O sistema é assíncrono com detectores de falhas. Um processo pode estar em um de dois estados: falho ou sem-falha. O modelo de falhas é o de parada com recuperação (*crash-recovery*). Os canais de comunicação são confiáveis e FIFO. Destaca-se que mensagens recebidas concorrentemente de diferentes emissores são entregues em qualquer ordem. Uma execução é uma sequência alternada de eventos e estados de processos. O efeito de um evento no estado de um processo conduz o processo a um novo estado. A ordem parcial entre os eventos é obtida através da relação *happened-before* [Kshemkalyani and Singhal 2011]. Os eventos são classificados em determinísticos e não-determinísticos.

Os eventos não-determinísticos de um processo são identificados e armazenados em determinantes. Um determinante de um processo i é formado pelo seu identificador, o identificador do emissor da mensagem e o tipo de evento (por exemplo, se uma recepção ou verificação de mensagem recebida), ou seja: $det_i \leftarrow (det_{id}, id_j, event\_type)$. Inicialmente, det_{id} é definido em 0 e é incrementado a cada novo determinante armazenado. Os processos têm acesso a um serviço de *event logger* (definido na Seção 3.3) para salvar, remover e recuperar os determinantes. Ao salvar um determinante, o *event logger* retorna ao processo o identificador do determinante. Um processo i incrementa contadores $sent_i[j]$ e $recv_i[j]$ ao enviar ou receber uma mensagem do processo j, respectivamente. Os contadores são representados por vetores de tamanho $|V|$, onde cada posição no vetor é indexada pelo identificador do processo correspondente e inicializada em 0. O protocolo assume a abordagem *sender-based* [Johnson and Zwaenepoel 1987]: toda vez que o processo i envia uma mensagem msg ao processo j, uma cópia da mensagem é mantida na memória volátil de i. O identificador da mensagem é formado pelo id do emissor e a sequência de envio, ou seja $msgs_i \leftarrow (id_j, sent_i[j], msg)$.

3.2. Descrição do Protocolo

Periodicamente, usando o seu relógio local, cada processo realiza um *checkpoint*, o qual é armazenado em um dispositivo confiável. O *checkpoint* inclui o estado do processo, as suas mensagens enviadas mantidas em memória volátil, o identificador do

último determinante submetido ao *event logger* e os contadores de mensagens enviadas e recebidas. Ou seja, o *checkpoint* em um processo i tem o seguinte formato: $ck_i \leftarrow (p_i, msgs_i, det_{id}, sent_i[], recv_i[])$. Apenas o último *checkpoint* do processo é mantido. A cada *checkpoint* em i, todos os determinantes de i mantidos até então no *event logger* podem ser removidos. Além disso, cada processo j pode apagar as mensagens enviadas para i mantidas em sua memória tal que $sent_j[i] \leq recv_i[j]$.

A recuperação de um processo ocorre da seguinte forma. Um novo processo i é criado para substituir o processo falho. O processo i lê o seu *checkpoint*, obtém os seus determinantes do *event logger* e solicita a cada processo j que reenvie as mensagens de i a partir da última mensagem recebida por i, ou seja, $\forall j \in V$, i envia $recv_i[j]$ a j. Então, j reenvia msg tal que $sent_j[i] > recv_i[j]$. O processo j também envia $recv_j[i]$ para que i saiba qual foi a última mensagem que j recebeu de i. As mensagens $sent_i[j] \leq recv_j[i]$ não serão reenviadas, pois j as recebeu antes da falha de i.

O processo i então reinicia sua computação a partir do seu *checkpoint*. Cada evento determinístico encontrado será reproduzido. Os eventos não-determinísticos serão substituídos pelas informações contidas nos determinantes lidos do *event logger*. Por exemplo, considere a falha de P_1 (Figura 1). Antes de falhar, os determinantes det_1, det_2 e det_3 correspondentes às mensagens m_2, m_4 e m_3 foram salvos no *event logger*. Em sua execução normal, P_1 aguarda mensagens de quaisquer emissores. Mas, em recuperação, e de posse dos determinantes det_1, det_2 e det_3 recuperados do *event logger*, P_1 aguarda as mensagens vindas de P_2, P_2 e P_0, nessa ordem. Uma vez que cada mensagem é reaplicada, a mensagem m_5 é então gerada e armazenada em memória volátil. A partir de então, a execução de P_0, P_1 e P_2 segue normalmente.

3.3. O Serviço Proposto de *Event Logger* Baseado em Consenso

O serviço de *event logger* proposto é baseado em consenso. O consenso é uma abstração fundamental na computação distribuída tolerante a falhas. O consenso pode ser usado para construir um serviço de *event logger* usando a abordagem de máquina de estado [Schneider 1990]. A replicação máquina de estado regula como os comandos devem ser propagados e executados pela réplicas para que o serviço seja consistente. No nosso caso, os comandos são solicitações para salvar um determinante, propagados e executados pelas réplicas do *event logger*. A propagação dos comandos possui dois requisitos: (i) cada réplica sem-falha deve receber cada comando e (ii) duas réplicas quaisquer não podem discordar na ordem dos comandos recebidos e executados. Se a execução é determinística, então as réplicas alcançarão o mesmo estado e produzirão a mesma saída ao executar a mesma sequência de comandos.

Paxos é um algoritmo de consenso que garante a propriedade de segurança do consenso em um sistema assíncrono sujeito a falhas e a propriedade de progresso sob suposições de assincronia fraca. No Paxos os processos podem assumir os seguintes papéis distintos: *proposers*, *acceptors* e *learners*. Os *proposers* propõe um valor, os *acceptors* escolhem um valor e os *learners* aprendem o valor decidido. Um único processo pode assumir qualquer uma dessas funções, e múltiplas funções simultaneamente. Paxos é ótimo em termos de resiliência [Lamport 2006]: para tolerar f falhas ele requer $2f + 1$ *acceptors*. Um processo *coordenador* ainda pode ser escolhido para evitar que os *proposers* concorram indefinidamente por um valor. Com a presença de um coordenador, um comando pode ser aprendido em três passos de comunicação [Lamport 2001].

3.3.1. *Event Loggers* Baseados em Paxos Clássico e Paxos Paralelo

Dois procolos baseados em Paxos Clássico e Paxos Paralelo são propostos para construir o *event logger* replicado. A Figura 2 apresenta as três abordagens.

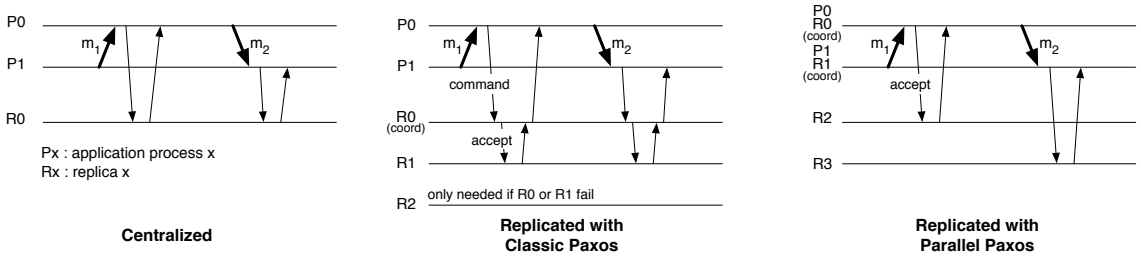


Figura 2. Três implementações de um *event logger*.

Os processos da aplicação são representados por $P0$ e $P1$. As mensagens m_1 e m_2 são mensagens trocadas entre os processos da aplicação. Cada mensagem recebida pelo processo da aplicação gera uma requisição ao *event logger*, representado por Rx em cada abordagem. A abordagem centralizada conta apenas com um único *event logger* (R_0) e assim não tolera falhas. As abordagens baseadas em Paxos contam com 3 *event loggers* replicados e assim podem tolerar uma falha (isto é, $f = 1$). Considera-se que os coordenadores do Paxos executaram previamente a primeira fase do protocolo e podem seguir com a segunda fase ao receber um comando.

O Paxos clássico baseia-se na replicação máquina de estado clássica. Os processos da aplicação são *proposers* e as réplicas de *event logger* são os *acceptors* e os *learners*. Cada processo da aplicação submete comandos ao coordenador, um processo entre os *acceptors* para registrar os determinantes. O coordenador recebe o comando, executa o Paxos para registrar o comando em um quórum de réplicas e responde ao processo da aplicação (ver Figura 2). Por exemplo, um determinante é recebido pelo coordenador (R_0), que também é um *acceptor*. O coordenador executa diretamente a segunda fase do Paxos. Ao receber a resposta de um quórum de *acceptors*, no caso R_0 e R_1, o consenso é finalizado e R_0 responde ao processo $P0$. Em "boas execuções" (isto é, na ausência de falhas de processos) um determinante é registrado em quatro passos de comunicação e $2f + 2$ troca de mensagens ponto-a-ponto. Em contraste, um *event logger* centralizado pode registrar eventos após dois passos de comunicação e duas trocas de mensagens.

No segundo protocolo, Paxos Paralelo, há um sequência separada de execuções do Paxos associada a cada processo da aplicação. Isso significa que cada processo possui seu próprio conjunto de réplicas o que permite certas otimizações. Primeiro, uma vez que cada processo tem sua própria sequência de execuções do Paxos, o processo não compete com outros processos da aplicação nas execuções de Paxos e assim não há necessidade de um único coordenador que recebe requisições de diferentes processos. Em boas execuções, o processo é o único *proposer* em sua sequência de Paxos. Uma segunda otimização é a seguinte: ao usar diferentes conjuntos de réplicas o desempenho não é mais limitado pelo o que o coordenador e os *acceptors* podem suportar. Terceiro, é possível co-alocar os processos da aplicação e o *acceptor*-coordenador no mesmo processador. Este esquema pode registrar um determinante após dois passos de comunicação e $2f$ mensagens.

Um *event logger* implementado com Paxos Paralelo apresenta o mesmo número

de passos de comunicação de um *event logger* centralizado, com a vantagem de tolerar um número configurável de falhas e apresentar desempenho escalável. Quando configurado para tolerar uma falha ($f = 1$), o número de trocas de mensagens é o mesmo de um *event logger* centralizado. Além disso, para economizar recursos, *acceptors* não co-alocados com os processos da aplicação podem ser hospedados em um mesmo nodo físico. Por exemplo, um único nodo pode hospedar todos esses *acceptors*. Nesse caso, o Paxos Paralelo usa o mesmo número de nodos requerido pelo abordagem centralizada.

Ao se recuperar de uma falha, um processo de aplicação deve recuperar todos os seus determinantes. Na abordagem centralizada, o processo da aplicação contacta o *event logger*. Com a abordagem replicada, isso é feito contactando um quórum de *acceptors*.

4. Implementação

Esta seção descreve as implementações do protocolo de registro de mensagens pessimista em MPI e dos *event loggers*. A implementação do protocolo de registro de mensagens proposto se apoia na especificação ULFM (*User Level Failure Mitigation*). A ULFM é o mais recente esforço do MPI-Fórum para padronizar a semântica de tolerância a falhas em MPI [Bland et al. 2013], pois, tradicionalmente, as aplicações MPI abortam toda a aplicação se um único processo falhar. Em [Gamell et al. 2014] é apresentado o esquema que o MPI em conjunto com a ULFM emprega para lançar novos processos ao detectar um processo falho. A comunicação entre os processos em MPI é FIFO e implementada através do protocolo TCP.

A implementação do protocolo de registro de mensagens foi encapsulada em uma biblioteca para facilitar o seu uso por uma aplicação MPI. A seguir destacamos brevemente as principais primitivas. A primitiva `framework_init()` inicia o interceptador de eventos (descrito mais adiante) e as estruturas de dados necessárias, como os contadores de mensagens enviadas e recebidas. Essa primitiva também define parâmetros como, por exemplo, a periodicidade do *checkpointing* e o seu local de armazenamento. No entanto, a sua principal função é classificar o processo MPI como um processo original ou um novo processo em recuperação. A primitiva `aloc_data()` define quais variáveis da aplicação serão incluídas no *checkpointing*. Se o processo MPI for um novo processo em recuperação, a primitiva `recovery()` é responsável por restaurar o *checkpoint* do processo, solicitar os seus determinantes do *event logger* e orquestrar a recuperação conforme definido na Seção 3. A primitiva `chkp()` é responsável por realizar o *checkpointing* do processo. A primitiva `garbage_colletor()` realiza a limpeza periódica das mensagens antigas mantidas em memória. A biblioteca foi programada em linguagem C.

A interface PMPI [MPI Forum 2015] é utilizada para interceptar as chamadas MPI de forma transparente para o desenvolvedor. Dessa forma, é possível inserir código antes e depois de cada chamada MPI. Os eventos não-determinísticos são detectados de acordo com o refinamento proposto em [Bouteiller et al. 2010]. Toda primitiva de recepção, como `MPI_Recv`, que aguarda uma mensagem de uma fonte qualquer (`MPI_ANY_SOURCE`) gera um evento não-determinístico. Assim também acontece com primitivas que verificam se há uma mensagem pronta para ser recebida, como a `MPI_Probe`. Além das primitivas de recepção, as primitivas não bloqueantes que inspecionam se o recebimento da mensagem foi concluído como, por exemplo, `MPI_Waitsome`, `MPI_WaitAny`, `MPI_Test` também geram eventos não-

determinísticos. Cada mensagem enviada é copiada para a memória e armazenada em uma tabela *hash*. A biblioteca khash[1] foi usada como implementação de uma tabela *hash*. A implementação PMPI foi realizada em C.

Um interceptador de eventos, implementado através de uma *thread*, é responsável por submeter e recuperar os determinantes do *event logger*. Ao iniciar, cada processo MPI instancia um interceptador que se conecta ao *event logger*. O interceptador pode ser configurado para submeter determinantes de forma síncrona ou assíncrona. No modo síncrono, após submeter um determinante, o processo da aplicação aguarda a confirmação do *event logger* antes de prosseguir. No modo assíncrono o processo da aplicação pode prosseguir antes de receber a confirmação. Por padrão, todos os resultados foram obtidos usando o modo síncrono. O protocolo pessimista permite atrasar o recebimento das confirmações até o envio de uma mensagem a outro processo. Essa otimização foi aplicada na implementação do protocolo proposto. Foram implementadas três versões de *event loggers*: centralizado, baseado no Paxos Clássico e baseado no Paxos Paralelo. O interceptador e os *event loggers* foram implementados em C usando a biblioteca libevent versão 2.022[2]. A biblioteca libpaxos[3] versão 3 foi utilizada para desenvolver os *event loggers* baseado em Paxos.

5. Resultados Experimentais

Os resultados experimentais avaliam os *event loggers* propostos, incluindo o procedimento de recuperação. Os *event loggers* baseados em consenso são comparados à implementação do *event logger* centralizado. A latência e a vazão, em termos do número de determinantes armazenados por segundo, para cada um dos *event loggers* são apresentados em quatro execuções MPI: a aplicação AMG[4] (Algebraic Multigrid Solver), os *kernels* LU (Lower-Upper Gauss-Seidel solver) e MG (Multi-Grid solver) do NAS *parallel benchmark* versão 3.2.[5], e o algoritmo de árvores de cortes de Gusfield[6]. O algoritmo de Gusfield também foi empregado para avaliar a recuperação. As aplicações foram executadas usando a implementação OpenMPI/ULFM 1.1[7]. Os experimentos foram conduzidos em um *cluster* dedicado que consiste de 40 nodos, cada um com dois processadores Intel(R) Quad-Core Xeon L5420 2.5 GHz e 8 Gbytes de RAM. Os nodos estão conectados através de uma rede *Gigabit Ethernet*.

5.1. Os *Event Loggers*

Primeiramente, o desempenho dos *event loggers* foram avaliados através de uma simples aplicação MPI onde cada processo submete um novo determinante ao *event logger* ao receber a confirmação de que o determinante submetido anteriormente está armazenado. Nessa aplicação, os processos MPI não trocam mensagens entre si e os determinantes têm o tamanho de 50 bytes. A Figura 3 apresenta a vazão e a latência (ambas em milissegundos) conforme o número de processos aumenta até 128. Nesse experimento há um nodo

[1] http://attractivechaos.awardspace.com/khash.h.html
[2] http://libevent.org/
[3] https://bitbucket.org/sciascid/libpaxos
[4] https://codesign.llnl.gov/amg2013.php
[5] https://www.nas.nasa.gov/publications/npb.html
[6] http://www.deinfo.uepg.br/ jcohen/parallel-cuttree.html
[7] https://bitbucket.org/icldistcomp/ulfm/downloads

dedicado que hospeda o *event logger* centralizado. O *event logger* baseado em Paxos clássico conta com um *proposer* em um nodo dedicado e três *acceptors* (isto é, $f = 1$) e três *learners*; cada *acceptor* e *learner* executa em 1 nodo dedicado. A configuração do Paxos Paralelo é a seguinte: cada sequência de Paxos usa um *proposer*, que também é *learner*, e três *acceptors* (isto é, $f = 1$). O *proposer/learner* está hospedado junto ao processo MPI. As sequências de *acceptors* estão em três nodos dedicados, com cada *acceptor* da sequência em um nodo. Os processos MPI estão executando nos 40 nodos do *cluster*.

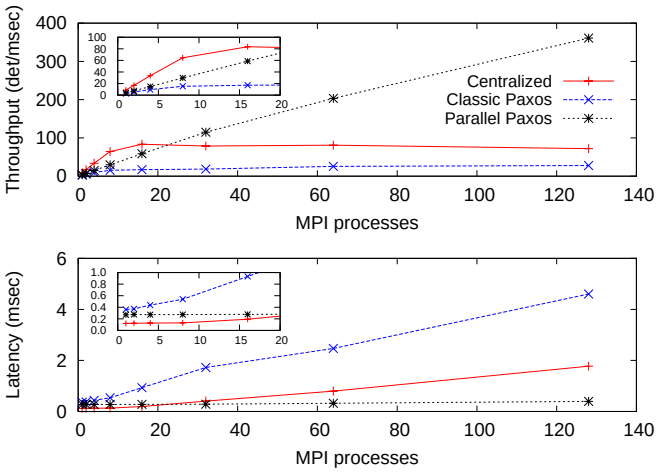


Figura 3. Vazão e latência para três as abordagens de *event loggers*.

A Figura 3 apresenta a vazão e latência para os três *event loggers* implementados. O pequeno gráfico interno é um zoom dos 20 primeiros processos. O *event logger* centralizado atinge a vazão máxima de 83 determinantes por milissegundo (det/ms) com 16 processos. O *event logger* baseado em Paxos Clássico alcança a vazão máxima de 28 det/ms com uma latência de 4,3 ms com 128 processos MPI. Já o *event logger* baseado em Paxos Paralelo nunca chega a saturar nesse experimento: a vazão aumenta proporcionalmente ao número de processos e a latência permanece aproximadamente constante, abaixo de 4 ms. Com 128 processos, a abordagem baseada em Paxos Paralelo tem 5 vezes a vazão da abordagem centralizada e uma latência muito menor. A seguir apresentamos resultados de desempenho dos *event loggers* coletados durante a execução de 4 aplicações MPI (Figura 4). A primeira barra corresponde ao desempenho da aplicação sem o uso de qualquer *event logger* e sem registrar qualquer tipo de evento. A segunda barra indica os resultados para a abordagem centralizada e as duas últimas para os *event loggers* baseados em Paxos Clássico e Paxos Paralelo, respectivamente. Para os três primeiros resultados (AMG, LU e MG) os *event-loggers* são configurados conforme descrito no início desta seção. Foram usados 16, 32, 64 e 128 processos MPI, exceto para o último gráfico (Algoritmo de Gusfield) que foi executado com 4, 8, 16 e 32 processos MPI.

A aplicação AMG possui o seguinte não-determinismo em seu código: todas as chamadas `MPI_Iprobe` usam a marcação `MPI_ANY_SOURCE` e somente uma chamada `MPI_Recv`, entre muitas, usa tal marcação. A aplicação também possui primitivas de verificação `MPI_Test` e `MPI_Testall`. Embora as primitivas de verificação gerem muitos eventos não-determinísticos, um determinante somente é gerado quando a mensagem está pronta para ser recebida. O número de vezes em que a mensagem não estava

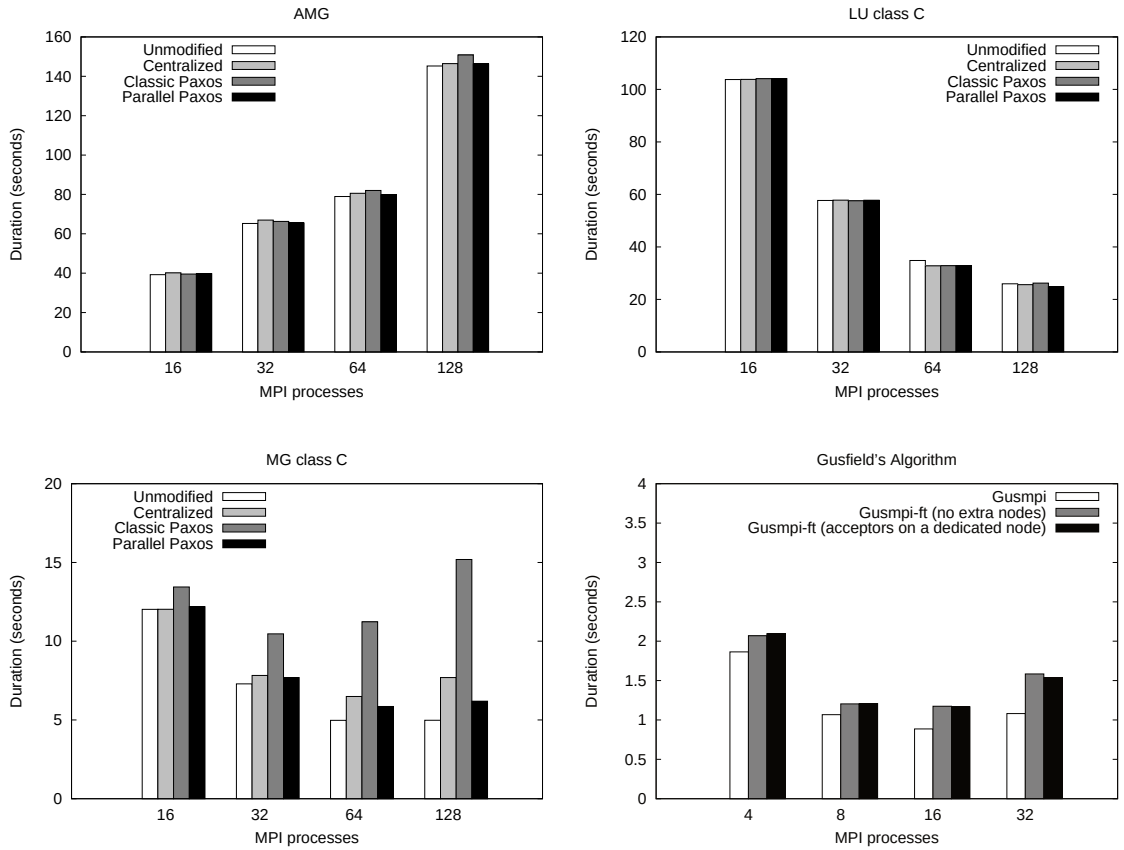


Figura 4. Quatro aplicações MPI usando os *event loggers* propostos.

pronta para ser recebida é incluído no determinante. Considerando todos os eventos gerados, os eventos não-determinísticos da aplicação AMG correspondem a menos de 1% do total. No entanto, há picos onde a vazão atinge até 70 det/ms. A Figura 4 (AMG) apresenta os resultados obtidos. O pior resultado foi obtido usando a abordagem do Paxos Clássico para 64 e 128 processos, houve uma sobrecarga de aproximadamente 3,8%. A abordagem centralizada apresentou uma sobrecarga de aproximadamente 2% para 16, 32 e 64 processos. O *event-logger* usando o Paxos Paralelo teve a menor sobrecarga, abaixo de 1,3% para todos os números de processos, sendo inclusive melhor que o centralizado.

Os *kernels* LU e MG são os únicos da versão 3.2 do NAS que apresentam eventos não-determinísticos. Em ambos, os únicos eventos não-determinísticos são `MPI_Recv` com a marcação `MPI_ANY_SOURCE`. No NAS, o tamanho das entradas para os experimentos é classificado em classes. Resultados da classe C são apresentados porque como o tamanho da entrada da classe C é menor do que da classe D, a taxa de comunicação é maior e, consequentemente, maior é a carga nos *event-loggers*. Enquanto o *kernel* LU gera menos de 1% de eventos não-determinísticos, no kernel MG quase 100% dos eventos são não-determinísticos, considerando somente as recepções. Embora as aplicações MG e AMG resolvam problemas similares, a MG possui muito mais não-determinismo em seu código. Na verdade, o não-determinismo no código é frequentemente uma escolha do desenvolvedor. Conforme apresenta a Figura 4, os *event-loggers* virtualmente não impactam no desempenho do LU. Ao contrário, o desempenho do MG é impactado pelos *event loggers*. Enquanto a abordagem usando o Paxos clássico apresenta uma sobrecarga

de 125% e 200% para 64 e 128 processos, o *event logger* centralizado mostra uma sobrecarga de 31% e 55% para a mesma execução. O *event logger* usando o Paxos Paralelo tem uma sobrecarga de 17,71% e 24,26% para 64 e 128 processos, respectivamente.

O algoritmo de Gusfield, último gráfico da Figura 4, foi executado com 4, 8, 12 e 32 processos MPI para uma entrada de 2000 vértices e 21.990 arestas. Esta é uma aplicação do tipo mestre-escravo. Somente o mestre gera eventos não-determinísticos, todas as suas recepções usam a marcação `MPI_ANY_SOURCE`. Nesse experimento foram avaliados a aplicação sem modificações e com a abordagem Paxos Paralelo. Porém, há duas configurações distintas do Paxos Paralelo. Na primeira, cada sequência de *acceptor* usa 3 nodos dedicados, como nos experimentos anteriores. Na segunda configuração, os *acceptors* são co-alocados com os processos da aplicação e, assim, não há nodo extra usado. Como é possível perceber, não houve diferença significativa de sobrecarga entre as duas configurações de Paxos Paralelo nesse experimento.

5.2. Recuperação da Aplicação

O protocolo proposto, incluindo a recuperação, foi avaliado usando o algoritmo de Gusfield. Todas as primitivas descritas na seção 4 são inseridas no código. O algoritmo foi levemente modificado para lidar com cenários de falha e recuperação. Basicamente, o algoritmo de Gusfield executa cálculos de fluxos máximos em um laço de repetição. O mestre solicita que seus escravos calculem uma parte do problema e ao receber o resultado de um escravo envia uma nova parte do problema a esse escravo. A primitiva `chkp()`, por exemplo, foi inserida no fim do laço de repetição. Já a primitiva `recovery` é inserida antes do laço. Foram incluídos no *checkpointing*, entre outros, os vetores que guardam os valores calculados a cada iteração.

A Figura 5 apresenta a execução do algoritmo para 4, 8, 16, 32 e 64 processos em uma entrada com 1000 vértices e 99.900 arestas. O *event logger* baseado na abordagem Paxos Paralelo foi empregado na execução. A primeira barra corresponde à execução do algoritmo original, isto é, sem modificações. A segunda barra apresenta a sobrecarga causada pelo registro dos determinantes no *event logger*. A terceira barra diz respeito a um cenário de falha e recuperação do mestre. Nessa execução houve a inserção de falhas aleatórias durante a execução. A quarta barra se refere à falha e recuperação dos escravos. Durante a execução um dos escravos falhava aleatoriamente. A quinta barra representa a adição ao cenário de falha do mestre de um *checkpointing* durante a metade da execução. A sexta barra adiciona a esse último cenário a coleta de mensagens antigas, configurado para executar pelo menos uma vez durante a execução.

De acordo com o gráfico da Figura 5 é possível perceber que o tempo que a aplicação leva para se recuperar de forma consistente usando o protocolo proposto não é significante. Lembrando que o processo de recuperação inclui lançar um novo processo, recuperar os seus determinantes do *event logger*, ler o seu *checkpoint* e refazer a computação perdida. Tendo como base a aplicação sem qualquer modificação (primeira barra da Figura 5), quando o mestre se recupera de uma falha (terceira barra), a maior sobrecarga observada foi de 15,16% para 32 processos. Porém, quando o mestre realiza um *checkpointing* (quinta barra), a sobrecarga para esse mesmo caso cai para 13,2%. O mesmo ocorre no cenário de 64 processos, a sobrecarga com o mestre se recuperando cai de 9,42% para 8,62%. Ou seja, o tempo que o mestre leva realizar um *checkpoint* é compensado no processo de recuperação. O tempo de recuperação dos escravos (quarta barra)

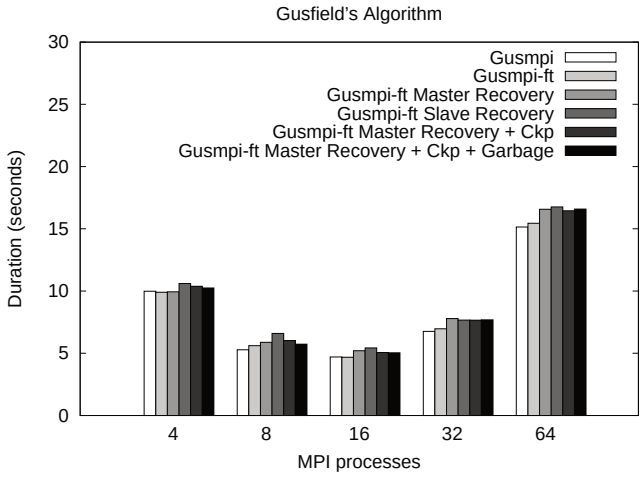


Figura 5. Recuperação do algoritmo Gusfield.

foi superior ao tempo de recuperação do mestre nesse experimento porque os escravos repetem a sua computação desde a última falha. Além disso, o mestre precisa se certificar que não deixou de receber mensagens enviadas por algum outro escravo devido a detecção da falha. É possível otimizar a implementação nesse cenário porque os escravos não precisam guardar um estado prévio. Nesse caso, ao detectar a falha de um escravo o mestre faria um *checkpoint* do seu estado e simplesmente lançaria um novo escravo que não precisaria refazer a sua computação. É possível observar também que ao adicionar a limpeza de mensagens antigas (última barra) não houve impacto significativo no tempo de execução (se comparado a penúltima barra).

6. Conclusão

Neste trabalhos apresentamos um protocolo pessimista para registro de mensagens construído com um *event logger* distribuído e tolerante a falhas baseado em consenso. Dois *event logger* baseados no algoritmo Paxos foram implementados e avaliados. Ao empregar o algoritmo de consenso Paxos, os *event logger* propostos garantem a propriedade de segurança mesmo se o sistema for assíncrono e o progresso apesar de falhas de processos. Nos experimentos realizados, usando as aplicações AMG, LU, MG e algoritmo de Gulsfield, o *event logger* baseado em Paxos Paralelo apresentou desempenho superior a abordagem centralizada. A implementação do protocolo foi encapsulada em uma biblioteca para facilitar o seu uso por uma aplicação MPI. O algoritmo de Gusfield foi levemente modificado empregando as primitivas implementadas. Resultados demonstram que perante falhas os processos da aplicação se recuperam eficientemente. Trabalhos futuros incluem avaliar o protocolo proposto em aplicações MPI de larga escala.

Referências

Bland, W., Bouteiller, A., Hérault, T., Bosilca, G., and Dongarra, J. (2013). Post-failure recovery of MPI communication capability: Design and rationale. *International Journal of HPC Applications*, 27(3):244–254.

Bouteiller, A., Bosilca, G., and Dongarra, J. (2010). Redesigning the message logging model for high performance. *Concurrency and Computation: Practice and Experience*, 22(16):2196–2211.

Bouteiller, A., Collin, B., Herault, T., Lemarinier, P., and Cappello, F. (2005). Impact of event logger on causal message logging protocols for fault tolerant mpi. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 97–97.

Bouteiller, A., Hérault, T., Krawezik, G., Lemarinier, P., and Cappello, F. (2006). MPICH-V project: A multiprotocol automatic fault-tolerant MPI. *International Journal of HPC Applications*, 20(3):319–333.

Bouteiller, A., Ropars, T., Bosilca, G., Morin, C., and Dongarra, J. (2009). Reasons for a pessimistic or optimistic message logging protocol in MPI uncoordinated failure, recovery. In *Cluster*.

Cappello, F., Guermouche, A., and Snir, M. (2010). On communication determinism in parallel HPC applications. In *ICCCN*.

Egwutuoha, I. P., Levy, D., Selic, B., and Chen, S. (2013). A survey of fault tolerance mechanisms and checkpoint/restart implementations for hpc systems. *The Journal of Supercomputing*, 65(3):1302–1326.

Elnozahy, Alvisi, Wang, and Johnson (2002). A survey of rollback-recovery protocols in message-passing systems. *CSURV: Computing Surveys*, 34.

Fagg, G. E. and Dongarra, J. (2000). FT-MPI: Fault tolerant MPI, supporting dynamic applications in a dynamic world. In *Recent advances in PVM and MPI*, LNCS.

Gamell, M., Katz, D., Kolla, H., Chen, J., Klasky, S., and Parashar, M. (2014). Exploring automatic, online failure recovery for scientific applications at extreme scales. In *SC*.

Johnson, D. B. and Zwaenepoel, W. (1987). Sender-based message logging. In *FTCS*.

Kshemkalyani, A. D. and Singhal, M. (2011). *Distributed Computing: Principles, Algorithms, and Systems*. Cambridge University Press, Cambridge, UK.

Lamport (2001). Paxos made simple. *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 32.

Lamport, L. (2006). Lower bounds for asynchronous consensus. *Distributed Computing*, 19(2):104–125.

Lifflander, J., Meneses, E., Menon, H., Miller, P., Krishnamoorthy, S., and Kale, L. V. (2014). Scalable replay with partial-order dependencies for message-logging fault tolerance. In *CLUSTER*.

Liu, X., Xu, X., Ren, X., Tang, Y., and Dai, Z. (2013). A message logging protocol based on user level failure mitigation. In *ICA3PP*.

MPI Forum (2015). Document for a standard message-passing interface 3.1. Technical report, University of Tennessee, http://www.mpi-forum.org/docs/mpi-3.1.

Ropars, T. and Morin, C. (2009). Active optimistic message logging for reliable execution of MPI applications. In *Euro-Par*.

Ropars, T. and Morin, C. (2010). Improving message logging protocols scalability through distributed event logging. In *Euro-Par*.

Schneider, F. B. (1990). Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(3):299.

Quality of Service of an Asynchronous Crash-Recovery Leader Election Algorithm

Vinícius A. Reis[1] **, Gustavo M. D. Vieira**[1]

[1]DComp – CCGT – UFSCar
Sorocaba, São Paulo, Brasil

angiolucci@gmail.com, gdvieira@ufscar.br

Abstract. *In asynchronous distributed systems it is very hard to assess if one of the processes taking part in a computation is operating correctly or has failed. To overcome this problem, distributed algorithms are created using unreliable failure detectors that capture in an abstract way timing assumptions necessary to assess the operating status of a process. One particular type of failure detector is a leader election, that indicates a single process that has not failed. The unreliability of these failure detectors means that they can make mistakes, however if they are to be used in practice there must be limits to the eventual behavior of these detectors. These limits are defined as the* quality of service (QoS) *provided by the detector. Many works have tackled the problem of creating failure detectors with predictable QoS, but only for crash-stop processes and synchronous systems. This paper presents and analyzes the behavior of a new leader election algorithm named NFD-L for the asynchronous crash-recovery failure model that is efficient in terms of its use of stable memory and message exchanges.*

1. Introduction

Fault-tolerant distributed systems are created by the aggregation of many non fault-tolerant computer systems in clusters, coordinated by fault-tolerant software. These commodity clusters are modeled as *asynchronous* systems where there are no bounds to the message transmission delays and processing time. As a consequence, it is very hard to assess if one of the processes taking part of a computation is operating correctly or has failed. To overcome this limitation, many distributed algorithms assume the stronger computational model of asynchronous processes augmented with unreliable failure detectors [Chandra and Toueg 1996]. More than just flagging failed processes, unreliable failures detectors capture in an abstract way timing assumptions necessary to the correct operation of many distributed algorithms [Lamport 1998, Chandra and Toueg 1996]. The unreliability of these detectors is key to this abstraction: errors can be made and failures are detected eventually, in a way that reflects the timing uncertainties intrinsic to asynchronous distributed systems.

Unreliable as they are, failure detectors are enough to design correct distributed algorithms [Chandra et al. 1996]. However, if they are to be used in practice there must be limits to the eventual behavior of these detectors. These limits are properties of the distributed system (processes and network links), of the failure detector algorithm used and of its operational parameters, defining the *quality of service (QoS)* provided by the detector [Chen et al. 2002]. The QoS of a failure detector can have a direct impact on

the performance of a distributed algorithm. For example, the Paxos algorithm [Lamport 1998] uses a single coordinator process as the sequencer that orders and distributes a set of totally-ordered messages among a group of processes. The selection of the coordinator is made by a failure detector and the progress of the algorithm is halted while the coordinator is failed. If this coordinator is *wrongly* assumed to be failed by the failure detector, Paxos throughput is affected, even if the mistake is eventually corrected. The more mistakes the failure detector makes, the worse Paxos real world performance will be. Thus, QoS of failure detectors is an important parameter to be considered.

Many works have tackled the problem of creating failure detectors with predictable QoS. The seminal work of Chen et al. [Chen et al. 2002] defined a set of QoS metrics, created a new failure detector with a precise model of its QoS and used this model to configure its parameters to match a desired QoS. Other works have experimentally studied the QoS of failure detectors [Falai and Bondavalli 2005, Nunes and Jansch-Porto 2004], created more robust QoS models [Sotoma et al. 2006] and proposed application-specific QoS for a system-wide failure detector service [Hayashibara et al. 2004]. All these algorithms and QoS models were created assuming the crash-stop process abstraction, where processes can only fail by crashing and once crashed they never return to the computation. The only exception is the work of Ma et. al. [Ma et al. 2010] that tackles the problem of analyzing the QoS of Chen's algorithm in the crash-recovery process abstraction, where processes fail by crashing but later recover preserving their stable memory.

The works regarding QoS of failure detectors also share another, rather ironical, property: many assume a *synchronous* system where there is a (possibly unknown) bound to the message transmission delays and processing times. In this model it is easier to create models of QoS, usually based on the existence of synchronized clocks. Despite being a reasonable assumption in current systems, take for example the use of NTP in clusters, these algorithms are burdening the system designer with a more stringent synchrony model than the one required by the algorithm itself. One exception is one of the algorithms proposed by Chen et. al. [Chen et al. 2002], which in its turn has weaker QoS model based on the accuracy of a *predictor*. For asynchronous systems, the ability of the failure detector to ascertain the state of a process based on the history of its past communications is very important for its QoS, as shown by [Nunes and Jansch-Porto 2004] and explored by [Hayashibara et al. 2004]. Thus, there is no robust QoS model for asynchronous systems beyond the model originally proposed by Chen et al..

Moreover, no previous work has analyzed the QoS of a leader election algorithm. A leader election is a type of failure detector that instead of indicating if a single process has failed, it indicates a single process that has *not* failed, with the identity of this process agreed by all correct processes. In some senses, this type of failure detector consumes less system resources because it monitors only a single process [Larrea et al. 2000] and it is the weaker form of failure detector required to solve consensus [Chandra et al. 1996]. Paxos, a consensus algorithm, requires a leader election for asynchronous systems composed by crash-recovery processes. There are suitable leader election algorithms that support crash-recovery processes [Martín et al. 2009], but unfortunately there are no known algorithms with these properties that have a suitable model for their QoS.

This paper presents a new leader election for the asynchronous crash-recovery failure model that is efficient in terms of its use of stable memory and message exchanges.

Our algorithm named NFD-L is an extension of Chen's algorithm and retains its QoS properties in the absence of failures. We analyze the QoS of this algorithm and present experimental data as a first step to providing a complete QoS model for this failure detector in the presence of failures and recoveries.

2. Preliminaries

In this section we will describe the basic concepts of distributed systems and the relationship between its abstractions. We then use those abstractions to present more sophisticated concepts, such as failure models and failure detectors.

2.1. Synchrony and Failure Models

Asynchronous distributed systems have no bounds for (i) how much time it takes for a message to be delivered and (ii) how much time it takes for a process to do some computation. *Synchronous* distributed systems can rely on hard bounds for message delay and computing time. *Partially synchronous* distributed system are systems where processes and links behaves most of the time asynchronously, but there is an unknow time in the future after which it behaves synchronously [Cachin et al. 2011].

Another property that defines a distributed system is process failure models. The *crash-stop* model considers *correct* a process that never crashes. Once crashed, a process is considered *faulty* and it never recovers. The *crash-recovery* failure model considers correct a process that never crashes or crashes and recovers a finite number of times. Thus, in the *crash-recovery* model a faulty process is a process that crashes and never recovers or a process that crashes and recovers infinitely [Cachin et al. 2011].

In this paper we consider asynchronous processes augmented with unreliable failure detectors. This is a type of partially synchronous distributed system where the timing assumptions are confined to the failure detector. We also assume crash-recovery processes that communicate with each other by exchanging messages through links. The links may drop or delay messages, but only deliver a message previously sent by some process.

2.2. Failure Detection and Leader Election

Given a distributed system composed by processes and links, a failure detector is a component that outputs not necessarily correct information about which processes are correct or faulty. It works as a local component, queried by processes in order to know about which processes are still correct. When a process q queries its local failure detector about the state of process p, the only two possible responses it can receive are either *trust*, if the failure detector trusts process p is correct or *suspect*, if the failure detector suspects p is faulty. Failure detectors are usually implemented by exchanging messages through its links [Chen et al. 2002]: a process p sends a heartbeat message to another process q every η time units, if q receives no messages from p after a timeout plus a safety margin α, the failure detector in q will start suspecting p may be crashed.

An important result is that it is possible to build reliable distributed systems on top of an unreliable failure detector [Chandra and Toueg 1996]. It means a failure detector is not supposed to be correct while the system behaves asynchronously [Fischer et al. 1985], it may make mistakes by suspecting correct processes or trusting faulty ones. However, assuming a partially synchronous system, eventually the processes and links will behave

synchronously and then the failure detector will stop making mistakes. Based on these assumptions, reliable distributed systems are designed to be *safe* when the system behaves asynchronously and only *progress* when it behaves synchronously [Guerraoui 2000]. Despite the mistakes it can make, a failure detector is a powerful abstraction as it encapsulates unpredictable system behavior.

A failure detector is specified in terms of two properties: *completeness* and *accuracy* [Chandra and Toueg 1996]. Completeness is the property that describes how well a failure detector will perceive real failures, while accuracy is the property that describes how well it will avoid mistakes [Reynal 2005], e.g., false detections. Failures detectors will behave differently and support other failures models by simply strengthening or loosening completeness and accuracy [Guerraoui 2000].

Of special interest for this work is the Ω failure detector, originally presented in [Chandra et al. 1996]. The Ω failure detector is a *leader election*, a failure detector that outputs a single trusted process. Formally, the Ω failure detector is specified by the following properties [Guerraoui 2000]:

- *Eventual Accuracy*: There is a time after which every correct process trusts some correct process.
- *Eventual Agreement*: There is a time after which no two correct processes trust different processes.

This pair of properties ensures every correct process will eventually trust the same correct process. The eventual behavior means it is necessary a long enough period of synchrony in order for the properties to be achieved. The Ω failure detector is used as a building block to solve more complex problems such as consensus [Lamport 1998] and atomic broadcast [Chandra et al. 1996].

It is rather easy to create a leader election using regular failure detection. Let each process in a distributed system use a failure detector to monitor every other process and create a set C of processes it believes to be correct. Eventually the set C will be the same in all correct processes and the leader can be chosen as the process in C with the smallest pid. However, this reduction is very costly in terms of heartbeat messages, as it requires $N^2 - N$ heartbeat messages for each η, one for each unidirectional communication link. Moreover, this reduction doesn't provide a very useful property: *leader stability*. Leader stability is the ability of the failure detector to never remove the leadership from a correct process because another process (with a lower pid, for instance) begins to be trusted [Malkhi et al. 2005]. Leader stability is very useful for Paxos, for example, because a leadership change in this algorithm can be a costly operation [Vieira et al. 2014].

2.3. QoS of Failure Detection with Crash-Stop Failures

A set of failure detectors with a model for quality of service was first proposed by Chen et al. [Chen et al. 2002]. In that work, the authors presented the concept of QoS for failure detectors, alongside quantitative metrics to measure it. They also presented a set of failure detectors that were designed to have a precise model of the behavior of these metrics. Roughly speaking, quality of service means the failure detector is configured to meet strict application requirements and work accordingly to the network probabilistic behavior.

The failure detector proposed by Chen et al. has two main components: an *estimator* and a *configurator*. The estimator is responsible for analyzing the network probabilistic behavior in terms of message losses and message delays. The configurator is responsible for generating a suitable failure detector based on the behavior observed by the estimator and on the QoS requirements provided by the application developer. The application developer must specify the application requirements in terms of three metrics:

- Detection time (T_D): This measures how much time elapses from the occurrence of a crash and it being detected by the failure detector;
- Mistake recurrence time (T_{MR}): This measures the time between two consecutives mistakes made by the failure detector;
- Mistake duration (T_M): This measures how much time the failure detector takes to correct itself once it has made a mistake.

The result is a customized failure detector in which both heartbeat inter-sending interval η and the safety margin α are shaped to meet the given constraints, when it is feasible. Once configured, a failure detector is ready to work as usual: a monitored process p sends every η time units a heartbeat message to a monitor process q, which will wait for those messages for a specific time, plus a safety margin α.

In this paper we will discuss only one of the failure detectors proposed by Chen et al., namely the *New Failure Detector with Expected Arrival Time Calculation (NFD-E)*. The NFD-E algorithm is the most suitable for partially synchronous distributed systems as it assumes no clock synchrony among the processes and estimates the arrival time of future heartbeats. This algorithm is particularly interesting because its QoS model is based only on the variance of the message propagation delays and not on the unknown delays themselves. However there is some limitations about the required assumptions needed to this algorithm work properly. It assumes a crash-stop failure model and, although there is no assumption about synchrony among processes, it is necessary that each process has access to a local clock. Furthermore, it is assumed there is no clock drift between any two local clocks.

The NFD-E algorithm works as follows: for all $i \geq 1$, a monitored process p sends at time $i \cdot \eta$ a heartbeat message m_i to a monitor process q. Also for all $i \geq 1$, process q waits for message m_i until a *freshness point* $\tau_i = EA_i + \alpha$. If no message with a label equals or grater than i is received by q until τ_i expires, i.e., no fresh message arrives before its timeout, then q starts suspecting p. To estimate each arrival time EA_i, process q uses a *predictor* based on the set of n previously received messages from p. Let $m'_1, ..., m'_n$ be the messages received by q, $s_1, ..., s_n$ be the sequence numbers of such messages, $A'_1, ..., A'_n$ be their receipt time according to q's local clock and ℓ the largest sequence number among all $s_1, ..., s_n$. Then $EA_{\ell+1}$ is estimated by [Chen et al. 2002]:

$$EA_{\ell+1} \approx \frac{1}{n}\left(\sum_{i=1}^{n} A'_i - \eta s_i\right) + (\ell + 1)\eta \quad (1)$$

The predictor that calculates EA_i actually estimates the average message delay for the last n messages, shifting backward in time by $\eta \cdot s_i$ each A'_i. By adding α to estimation made by the predictor, the failure detector is able to absorb some variation in message

delay. A study made by [Nunes and Jansch-Porto 2004] observed that the predictor used in the NFD-E algorithm is not the more accurate, but the use of a constant safety margin α allows the failure detector to achieve a good QoS. Hence the predictor used by the NFD-E algorithm plays a more important role in QoS achievement than the one originally observed by its authors, being responsible for accounting for the uncertainty in message propagation delays typical of partially synchronous system.

The full algorithm as presented by its authors in [Chen et al. 2002] is shown in Algorithm 1.

Algorithm 1 NFD-E Algorithm

1: **procedure** SENDHEARTBEAT ▷ Procedure exclusive for p, using p's local clock.
2: **for** all $i \geq 1$, at time $i \cdot \eta$ **do** send heartbeat m_i to q
3:
4: **procedure** INITIALIZATION ▷ Procedures exclusives for q, using q's local clock
5: $\tau_0 \leftarrow 0$
6: $\ell \leftarrow -1$
7:
8: **upon event** $\tau_{\ell+1} = now()$ **do**
9: $output \leftarrow Suspect$
10: **upon event** receives message m_j at time t **do**
11: **if** $j > \ell$ **then**
12: $\ell \leftarrow j$
13: $\tau_{\ell+1} \leftarrow EA_{\ell+1} + \alpha$
14: **if** $t < \tau_{\ell+1}$ **then**
15: $output \leftarrow Trust$
16:

2.4. QoS of Failure Detection with Crash-Recovery Failures

In [Ma et al. 2010] the authors analyze the QoS of the synchronous *New Failure Detector with Synchronized Clocks (NFD-S)* algorithm found in [Chen et al. 2002]. This algorithm assumes a synchronous system and Ma et al. assume some sort of time synchronization to be present, such as the NTP protocol.

The expanded model outlined in [Ma et al. 2010] presented additional QoS metrics that complement those in [Chen et al. 2002] listed in Section 2.3:

- Query Accuracy Probability (P_A): it measures the probability that, at any arbitrary time when queried, the failure detector correctly indicates the state of the monitored process.
- Recovery Detection Time (T_{DR}): it measures the time the failure detector perceives a recovery at the monitored process.
- Detected Failure Proportion (R_{DF}): it measures the ratio of the detected crashes over the actual crashes.

The configuration of the failure detector and parameter estimation for a desired QoS are shown in [Ma et al. 2010]. While an interesting expansion of previous work,

[Ma et al. 2010] conclude that the proposed algorithm and QoS model needs to be enhanced, specifically to deal with short failures that may not be detected. Additionally, the experimental results in some tests differs substantially from the expected behavior of the implemented failure detector (specially for the R_{DF} QoS metric), reinforcing the need for more research for a failure detector in this system model. Because of these limitations, in this paper we do not use the expanded set of metrics for a crash-recovery system, but instead we use the basic set proposed by [Chen et al. 2002].

3. Algorithm

In the previous sections, we provided the theoretical background needed to understand key concepts such as synchrony models, failure detectors with quality of service and leader election. In this section we present the *New Failure Detector as a Leader Election Algorithm (NFD-L)* leader election algorithm, which is a derivative of NFD-E algorithm [Chen et al. 2002], shown in Algorithm 2. NFD-L is an efficient leader election with stability that only requires a single message each η. More importantly, NFD-L is designed to work on the crash-recovery system model with no clock synchrony among processes and requiring only a single stable memory write during process initialization.

3.1. A Failure Detector with QoS as a Leader Election

A leader election is a failure detector that outputs a single trusted process. The output of NFD-L is the pid of the trusted $leader$. It is possible that each process of the distributed system sees a different process as a $leader$ at the same time, but when the system behaves synchronously for a sufficient amount of time, eventually all correct processes will agree on the same $leader$. To achieve this we combine the general principle of the bully algorithm [Garcia-Molina 1982] with the failure detecting properties of the NFD-E algorithm, using process uptime as a priority measure to ensure a basic level of stability.

Each process starts (or recovers) knowing nothing about the current leader. If it receives no message from a leader process, it then assumes it is the leader and starts sending heartbeat messages. A process loses the leadership when it receives a message from a process with greater priority: greater $uptime$ or greater pid in the event of a tie on $uptime$. When a process recognizes another process as a leader, it stops sending heartbeats and starts monitoring heartbeats sent by the leader, in a behavior similar to NFD-E. Thus, at this moment, only a process acting as a leader sends periodic heartbeat messages to all other processes that will behave as monitors. The leader election stabilizes when each correct process receives the message of a single process with high enough priority.

In steady state operation and assuming a broadcast communication medium the cost of NFD-L will be equivalent to single instance of the NFD-E algorithm, compared to the $N^2 - N$ instances of the naive reduction. Furthermore, in the fail-free case after stabilization the QoS of NFD-L will be equivalent to the QoS of NFD-E as it is a generalization of this algorithm.

3.2. Dealing with Crashes and Recoveries

Each heartbeat message m sent by a monitored process p carries an incremental sequence number i used by a monitor process q to determine if an incoming heartbeat is still fresh,

Algorithm 2 NFD-L Algorithm

1: **procedure** INITIALIZATION
2: $leader \leftarrow \perp$
3: $self.uptime \leftarrow 0$
4: $retrieve(self.zerotime)$
5: **if** $self.zerotime = \perp$ **then**
6: *self.zerotime* $\leftarrow$ *now()*
7: *store(self.zerotime)*
8: $i \leftarrow \frac{now()-self.zerotime}{\eta}$
9: **procedure** SENDHEARTBEAT
10: **if** $self.pid = leader$ **then**
11: **for** all $i \geq 1$, at time $i \cdot \eta$ **do**
12: send heartbeat m_i to all processes
13: $self.uptime \leftarrow self.uptime + 1$
14:
15: **upon event** receives message m_j at time t **do**
16: **if** $sender(m_j) = leader$ **then**
17: **if** $j > \ell$ **then**
18: $\ell \leftarrow j$
19: $\tau_{\ell+1} \leftarrow EA_{\ell+1} + \alpha$
20: **else**
21: **if** $uptime(sender(m_j)) > uptime(leader)$ **then**
22: $\ell \leftarrow j$
23: $leader \leftarrow sender(m_j)$
24: $output \leftarrow leader$
25: **else**
26: **if** $uptime(sender(m_j)) = uptime(leader)$ **then** ▷ Tiebreaker
27: **if** $pid(sender(m_j)) > pid(leader)$ **then**
28: $\ell \leftarrow j$
29: $leader \leftarrow sender(m_j)$
30: $output \leftarrow leader$
31:
32: **upon event** $\tau_{\ell+1} = now()$ **do**
33: $leader \leftarrow self.pid$
34: $output \leftarrow leader$

thus trusting or suspecting p to remain as leader. After recovering from a crash, it is important that process p starts sending correct sequence numbers to q, i.e., process p should remember which one was the exact sequence number i it sent before it crashed. If process p "forgets" its sequence number and starts sending i from the initial value, it will not be trusted by process q until i reaches the value it held before process p crashed.

The naive solution to remember the correct sequence number i to be used after a crash, is to write on persistent storage i every time process p sends a heartbeat message m_i. However, while correct, this solution isn't efficient regarding the number of writes to costly stable memory. We propose a cheaper solution that uses a single write operation once a process is initialized and to a read operation per process recovery.

Our solution takes advantage of the assumption that every process has access to a local clock and that the i-th heartbeat message should be sent at time $i \cdot \eta$. It works as follows: when a process starts for the very first time or it recovers from a previous crash, it checks if it wrote on persistent storage a timestamp of a previous initialization. If there is not a previous timestamp on persistent storage, the process then writes the current timestamp on persistent storage. This way, this value will be read and constant at every single recovery of that process. Using this value a process calculates what the current message label i is supposed to be, based on a function of the elapsed time since the first startup and η (Line 8). It ensures that every time a process starts or recovers, the message label i will not violate the properties defined in [Chen et al. 2002], i.e., every message label i sent by a process will be greater than the previous label $i - 1$. The only requirement is that the local clock keeps increasing with no drift during crashes.

4. QoS Analysis

In this section we present the QoS analysis of the NFD-L algorithm. The QoS analysis is made using the three metrics proposed in [Chen et al. 2002], T_D, T_{MR} and T_M (Section 2.3), with the addition of the T_{DR} metric proposed in [Ma et al. 2010] for the crash-recovery failure model (Section 2.4). These metrics will be grouped in failure detector accuracy metrics (T_{MR}, T_M) and failure detector speed metrics (T_D, T_{DR}). The experiments have the objective of assessing if the QoS of the proposed failure detector is within the bounds achieved by Chen's algorithm.

4.1. Enviroment and configuration of the leader election

These experiments ran at the Maritaca computational cluster, located at Universidade Federal de São Carlos, Sorocaba, São Paulo. We used 5 identical nodes, each one equipped with an 8 cores/16 threads processor and 16 GB of volatile memory. Each node used a hard disk as its persistent storage and all nodes were interconnected by a Gigabit Ethernet link.

The NFD-L algorithm by design selects a node as the leader and the other four nodes behave as monitor processes, receiving heartbeats sent by the leader. We monitored the leader election output of each monitor process to assess the QoS metrics. To generate a constant load on the network and on the system, we used the leader election as a service provider for the Treplica replication framework [Vieira and Buzato 2008]. The replication rate was set to the saturation point of the application with this setup, which was 2700 operations per second on average.

To generate the parameters η and α for the leader election, we used the configurator described by [Chen et al. 2002] during a period when the network was under high load due to the test application. We then measured the network probabilistic behavior for 1 hour, repeating it 3 times in total in different hours of the day. The observed message loss probability $p_L = 0.0175917$ and the observed message delay variance $V(D) = 25.3356$ were then used with the following QoS metrics $T_D = 1000$ ms, $T_{MR} = 3600000$ ms and $T_M = 1000$ ms to obtain $\eta = 330$ ms and $\alpha = 670$ ms.

We made two set of experiments in order to respectively observe the accuracy and the speed of the failure detector. The accuracy was measured by counting the average mistake rate (T_{MR}) and the mistake duration (T_M) for each monitor process during 6 runs of 1 hour each. During the experiment the leader process did not fail, thus all suspicions were mistakes that were eventually corrected. Let $t_{m0}, ..., t_{mn}$ be the timestamps of the mistakes made by a process and $t_{r0}, ..., t_{rn}$ be the timestamps of the correction of those mistakes, then we computed the mistake rate of that process as:

$$\frac{1}{T_{MR}} = \frac{1}{\frac{1}{n}\left(\sum_{i=1}^{n}\left(t_{mi} - t_{mi-1}\right)\right)}$$

The average mistake duration of a process was computed as the average time taken to correct a mistake:

$$T_M = \frac{1}{n}\left(\sum_{i=0}^{n}\left(t_{ri} - t_{mi}\right)\right)$$

The speed of the leader election algorithm was measured by the crash detection time (T_D) and by the recovery detection time (T_{DR}). To be able to measure T_{DR}, we have randomly selected a node as a high priority leader, breaking the specification of NFD-L in such way that this leader would be immediately reelected after a recovery. We then injected a single crash in the leader process and then we recovered it, measuring the time elapsed for each process to detect the crash and, afterwards, the time elapsed for each process detect the leader recovery. This experiment consisted of 10 repetitions of a crash-recovery cycle, with 60 seconds between a crash and the following recovery. Let t_c be the time of a leader crash and t_d the time of detection of that crash by a monitor process, the detection time T_D was obtained by calculating the difference of times:

$$T_D = t_d - t_c$$

In a very similar way, the recovery detection time was calculated considering the recovery time t_r and the detection time of that recovery t_{dr} by a process:

$$T_{DR} = t_{dr} - t_r$$

For this experiment we used the NTP protocol at the local network to synchronize the clocks of the nodes of the cluster. This procedure is not necessary for the algorithm to work properly, but it made easier to calculate time differences between the nodes. By using the NTP protocol we also observed that the average message delay between the nodes were a thousand of times smaller than the message intersending interval η intended to be used in our experiments, so we considered the message delay negligible.

4.2. Analysis

We plotted the QoS metrics observed in the experimental runs in boxplots. For each run the data points represent the metric as measured by each one of the four monitor process with respect to the current state of the actual leader. This amounts to a total of 24 points for the experiments assessing the accuracy of the failure detector and 40 points for the experiments assessing the speed of the failure detector. The expected value of the QoS metric used to configure the NFD-L algorithm is shown by the traced line in all plots. Numeric values for the data points represented in the figures are shown in Table 1, listing the values of the 1st, 2nd and 3rd quartiles.

Figure 1(a) shows the observed mistake rate. The majority of the processes (the darker line at the bottom of the image) made no mistakes, successfully achieving the required QoS of $T_{MR} = 3600000$ ms. However, there were outlier processes that did not achieve the required QoS. Among these, there is a single outlier that is not shown on figure, with a mistake rate of 33.34 mistakes/s. These outliers consist of a couple of mistakes in an one hour experiment, negatively affecting the average mistake recurrence time. In fact, the way the metric is computed (as proposed in [Chen et al. 2002]) increases the impact of each of the low number of mistakes. For example, the 33.34 mistakes/s outlier was created by one pair of mistakes 30 ms apart. The Figure 1(b) shows the observed mistake duration. The processes that made no mistakes were disconsidered. All the observed processes did meet the required QoS $T_M = 1000$ ms.

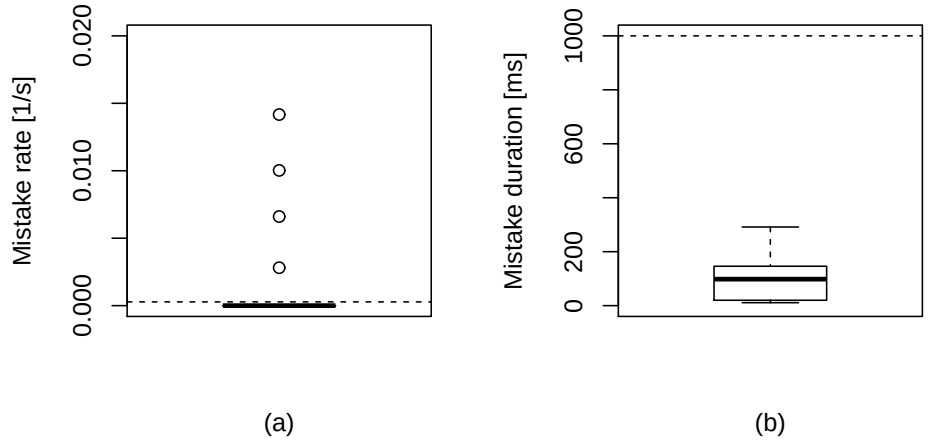


Figure 1. (a) The mistake rate $1/T_{MR}$ and (b) the mistake duration T_M. The traced line in the plots represents the upper bound QoS requirements for the leader election.

The crash detection time is shown in Figure 2(a). All the crashes were detected within the required QoS as the crash detection is bounded to $T_D = \eta + \alpha$. The Figure 1(b) shows the recovery detection time T_{DR} observed by the monitor processes. The speed metrics had less variation than the accuracy ones. It suggests that accuracy is more sensitive to process dependability than to network probabilistic behavior. As we assessed the QoS metrics of the failure detector integrated in a loaded application, pauses in the processing of the application created loss and delay of messages more intense than the ones

created by network probabilistic behavior alone. In a sense, this amounts to periods of omission faults where the processes stopped processing. We conjecture that a QoS model for the crash-recovery process model should be able to absorb this process behavior.

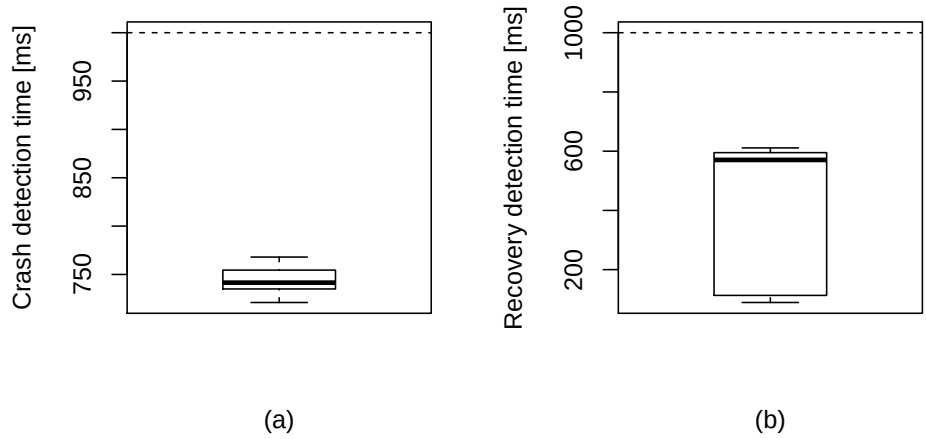


Figure 2. (a) The leader crash detection time T_D and (b) the leader recovery detection time T_{DR} observed by each monitor process. The traced line represents the upper bound QoS requirements for the leader election.

QoS Metric	1st. Quartile	Median	3rd. Quartile	Upper bound QoS
$1/T_{MR}$ (Figure 1(a))	0 1/s	0 1/s	0 1/s	$0.278 \cdot 10^{-3}$ 1/s
T_M (Figure 1(b))	20 ms	98.223 ms	146.03 ms	1000 ms
T_D (Figure 2(a))	735 ms	741.5 ms	754.25 ms	1000 ms
T_{DR} (Figure 2(b))	113 ms	570.5 ms	595 ms	1000 ms

Table 1. Summary of plot values

5. Conclusion

In this paper we presented the NFD-L leader election for the asynchronous crash-recovery failure model. This algorithm is an extension of the NFD-E [Chen et al. 2002] algorithm adapted to work as a leader election that is efficient in terms of its use of stable memory and message exchanges. The NFD-L algorithm used two novel techniques to achieve its efficiency: (i) instead of using a counter of crashes as usual [Martín et al. 2009], it uses a counter of uptime to prioritize stable processes and (ii) it uses a single write in stable memory to create a sequence of heartbeats that isn't interrupted by crashes, but only appears to be "paused", reducing crash failures to message omission failures.

We analyzed the performance of the NFD-L algorithm on a real system to verify how well it met the given QoS, configured as proposed in [Chen et al. 2002]. We were able to achieve the desired QoS, but the accuracy QoS metrics suffers for repetitive short period mistakes caused by the application. Our results suggest that accuracy of a failure detector is dependent on process dependability besides the effects of network probabilistic behavior. In particular, application overload spikes, lock contention and other performance

problems can severely interfere in the generation and processing of heartbeats. Thus, further enhancements of the QoS model are needed to make the algorithm suitable to an environment where the processes dependability is a main concern. We believe these changes must take in consideration crash-recovery parameters such as the ones found in [Ma et al. 2010] or be made more application specific as proposed in [Hayashibara et al. 2004].

Acknowledgments

We would like to thank Priscila Aiko Someda Dias for her valuable support during the statistical analysis of this work. This research was funded by the Brazilian *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES)* under the *Pró-Equipamentos* program (Edital 25/2011) and the *Demanda Social* program.

References

Cachin, C., Guerraoui, R., and Rodrigues, L. (2011). *Introduction to Reliable and Secure Distributed Programming*. Springer Berlin Heidelberg.

Chandra, T. D., Hadzilacos, V., and Toueg, S. (1996). The weakest failure detector for solving consensus. *J. ACM*, 43(4):685–722.

Chandra, T. D. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267.

Chen, W., Toueg, S., and Aguilera, M. (2002). On the quality of service of failure detectors. *Computers, IEEE Transactions on*, 51(5):561–580.

Falai, L. and Bondavalli, A. (2005). Experimental evaluation of the qos of failure detectors on wide area network. In *Proceedings of the 2005 International Conference on Dependable Systems and Networks*, DSN '05, pages 624–633, Washington, DC, USA. IEEE Computer Society.

Fischer, M. J., Lynch, N. A., and Paterson, M. S. (1985). Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382.

Garcia-Molina, H. (1982). Elections in a distributed computing system. *IEEE Trans. Comput.*, 31(1):48–59.

Guerraoui, R. (2000). Indulgent algorithms (preliminary version). In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '00, pages 289–297, New York, NY, USA. ACM.

Hayashibara, N., Defago, X., Yared, R., and Katayama, T. (2004). The φ accrual failure detector. In *Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems*, SRDS '04, pages 66–78, Washington, DC, USA. IEEE Computer Society.

Lamport, L. (1998). The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169.

Larrea, M., Fernández, A., and Arévalo, S. (2000). Optimal implementation of the weakest failure detector for solving consensus (brief announcement). In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '00, pages 334–, New York, NY, USA. ACM.

Ma, T., Hillston, J., and Anderson, S. (2010). On the quality of service of crash-recovery failure detectors. *IEEE Trans. Dependable Secur. Comput.*, 7(3):271–283.

Malkhi, D., Oprea, F., and Zhou, L. (2005). Ω meets Paxos: Leader election and stability without eventual timely links. In *DISC '05: Proceedings of the 19th International Conference on Distributed Computing*, volume 3724 of *Lecture Notes in Computer Science*, pages 199–213. Springer.

Martín, C., Larrea, M., and Jiménez, E. (2009). Implementing the omega failure detector in the crash-recovery failure model. *Journal of Computer and System Sciences*, 75(3):178–189.

Nunes, R. C. and Jansch-Porto, I. (2004). QoS of timeout-based self-tuned failure detectors: The effects of the communication delay predictor and the safety margin. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, DSN '04, pages 753–, Washington, DC, USA. IEEE Computer Society.

Reynal, M. (2005). A short introduction to failure detectors for asynchronous distributed systems. *ACM SIGACT News*, 36(1):53–70.

Sotoma, I., Roberto, E., and Madeira, M. (2006). A markov model for providing quality of service for failure detectors under message loss bursts. Technical Report IC-06-013, Institute of Computing, University of Campinas.

Vieira, G. M., Garcia, I. C., and Buzato, L. E. (2014). Seamless Paxos coordinators. *Cluster Computing*, 17(2):463–473.

Vieira, G. M. D. and Buzato, L. E. (2008). Treplica: Ubiquitous replication. In *SBRC '08: Proc. of the 26th Brazilian Symposium on Computer Networks and Distributed Systems*, Rio de Janeiro, Brasil.

Replicação Máquina de Estados Paralela e Reconfigurável

Alex Lobo[1] **, Eduardo Alchieri**[1] **, Fernando Pedone**[2] **,**
Fernando Dotti[3] **, Odorico Mendizabal**[4]

[1] Departamento de Ciência da Computação – Universidade de Brasília
[2] Faculdade de Informática – Universidade de Lugano
[3] Faculdade de Informática – Pontifícia Universidade Católica do Rio Grande do Sul
[4] Centro de Ciências Computacionais – Universidade Federal do Rio Grande*

Abstract. *State Machine Replication (SMR) is an approach used to implement fault-tolerant systems. In this approach, servers are replicated and client requests are deterministically executed in the same order by all replicas. Consequently, client requests must be ordered and sequentially executed by every replica. To improve system performance in multicore systems, parallel SMR allows parallel execution of requests, according to the degree of parallelism definied at startup. However, some requests still need sequential execution, impacting the system performance since additional synchronization is needed. This work proposes a protocol to reconfigure the degree of parallelism in parallel SMR. The protocol aims to improve performance by taking into account the workload. Experiments show the gains due to reconfiguration and shed some light on the behaviour of parallel SMR.*

Resumo. *A Replicação Máquina de Estados (RME) é uma abordagem muito utilizada na implementação de sistemas tolerantes a falhas. Esta técnica consiste em replicar os servidores e fazer com que os mesmos executem deterministicamente, e na mesma ordem, o mesmo conjunto de requisições. Para isso, as requisições devem ser ordenadas e executadas sequencialmente segundo esta ordem em todas as réplicas. Visando melhorar o desempenho do sistema em arquiteturas com múltiplos núcleos, RMEs paralelas tiram proveito da semântica das requisições e permitem a execução paralela de algumas delas, de acordo com um grau de paralelismo pré-definido. Porém, algumas requisições continuam precisando de execução sequencial e impactam negativamente o desempenho do sistema, visto que sincronizações adicionais são necessárias, de acordo com o grau de paralelismo. Este trabalho propõe um protocolo para RME paralela e com grau de paralelismo reconfigurável de acordo com o workload atual, visando tirar proveito em situação favoráveis e impactar o mínimo possível em situações desfavoráveis. Experimentos mostram os ganhos advindos com as reconfigurações e ajudam a elucidar o funcionamento deste tipo de sistema.*

1. Introdução

A Replicação Máquina de Estados (RME) [Schneider 1990] é uma abordagem muito utilizada na implementação de sistemas tolerantes a falhas [Lamport 1998, Schneider 1990, Castro and Liskov 2002]. Basicamente, esta técnica consiste em replicar os servidores e fazer com que os mesmos executem deterministicamente, e na mesma ordem, o mesmo

*Este trabalho recebeu apoio da CAPES através do projeto Scalable Dependability (88881.062190/2014-01) e do CNPq através do projeto FreeStore (457272/2014-7).

conjunto de operações requisitadas por clientes, fornecendo um serviço de replicação com consistência forte (*linearizability*) [Herlihy and Wing 1990].

Para manter o determinismo da execução, as operações são ordenadas e executadas sequencialmente seguindo a mesma ordem em todas as réplicas, o que limita o desempenho do sistema principalmente quando consideramos servidores atuais que possuem processadores com múltiplos núcleos, pois apenas um deles seria utilizado para a execução das operações. Com o objetivo de contornar esta limitação, recentemente surgiram abordagens que, tirando proveito da semântica das operações, empregam protocolos que suportam a execução paralela de algumas operações [Kotla and Dahlin 2004, Marandi et al. 2014, Marandi and Pedone 2014, Alchieri 2015, Zbierski 2015].

Estas abordagens, chamadas de RME paralelas, classificam as requisições em dependentes (ou conflitantes) e independentes (ou não conflitantes), de modo que as requisições independentes são executadas em paralelo nas réplicas. Já requisições dependentes devem ser executadas sequencialmente, o que exige alguma sincronização nas réplicas pois nenhuma outra operação pode ser executada paralelamente. Duas requisições são independentes quando acessam diferentes variáveis ou quando apenas leem o valor de uma mesma variável. Por outro lado, duas requisições são dependentes quando acessam pelo menos uma mesma variável e pelo menos uma das requisições altera o valor desta variável.

A quantidade de requisições independentes executadas em paralelo é configurada de acordo com o grau de paralelismo (número de *threads* de execução) definido na inicialização do sistema. Um número elevado aumenta o desempenho em *workloads* com requisições predominantemente independentes, mas impacta negativamente caso a predominância seja de operações dependentes, devido a necessidade de sincronizações adicionais. Por outro lado, um número baixo não tira proveito de *workloads* favoráveis (com predominância de requisições independentes) [Marandi et al. 2014, Alchieri 2015]. Desta forma, apesar de ser uma técnica muito promissora, configurar o grau de paralelismo é um problema complexo mesmo conhecendo-se o *workload a priori*, o que geralmente não é possível. Além disso, o *workload* geralmente sofre variações durante a execução do sistema [Le et al. 2016]. Com o objetivo de contornar estas limitações, as principais contribuições deste trabalho são:

- Proposta de um protocolo para RME paralela que melhora os protocolos anteriormente propostos, notadamente [Marandi et al. 2014, Marandi and Pedone 2014], por permitir a definição de grupos intermediários de dependências, além dos grupos de dependentes e independentes, aumentando o paralelismo na execução.
- Proposta de extensão do protocolo anterior para suportar reconfiguração no grau de paralelismo (número de *threads* de execução), possibilitando a adaptação ao *workload* atual, sendo que a principal vantagem desta técnica é que nenhuma informação preliminar se faz necessária.
- Apresentação e análise de uma série de experimentos realizados com uma implementação dos protocolos propostos, possibilitando uma melhor compreensão a respeito do funcionamento de uma RME paralela, bem como dos ganhos advindos com a sua reconfiguração.

O restante deste artigo está organizado da seguinte forma. A Seção 2 discute os conceitos envolvendo uma RME e as abordagens para melhorar seu desempenho. A Seção 3 apresenta os protocolos para RME paralela e reconfigurável. A Seção 4 analisa

alguns experimentos realizados. As conclusões do trabalho são apresentadas na Seção 5.

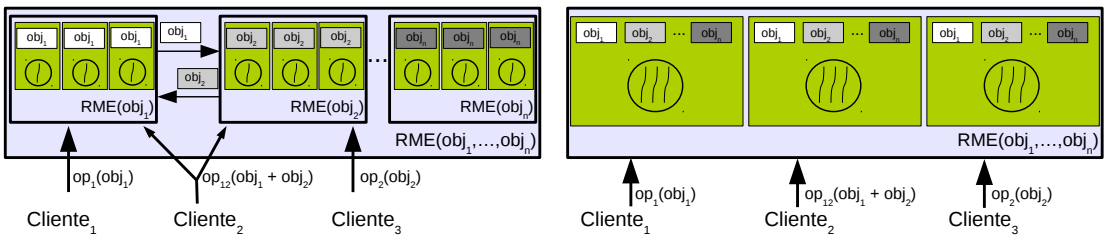
2. Replicação Máquina de Estados

A Replicação Máquina de Estados (RME) [Schneider 1990] é uma abordagem muito utilizada na implementação de sistemas tolerantes a falhas [Lamport 1998, Schneider 1990, Castro and Liskov 2002] e consiste em replicar os servidores e coordenar as interações entre os clientes e as réplicas, com o intuito de que as várias réplicas apresentem a mesma evolução em seus estados. São necessário $2f + 1$ ou $3f + 1$ réplicas para tolerar até f falhas por *crash* ou bizantinas, respectivamente.

Para que as réplicas apresentem a mesma evolução em seus estados, é necessário que: *(i)* partindo de um mesmo estado inicial e *(ii)* executando o mesmo conjunto de requisições na mesma ordem, *(iii)* todas as réplicas cheguem ao mesmo estado final, definindo o determinismo de réplicas. Para prover o item *(i)* basta iniciar todas as réplicas com o mesmo estado (i.e., iniciar todas as variáveis que representam o estado com os mesmos valores nas diversas réplicas), procedimento que pode não ser trivial se considerarmos a possibilidade de recuperação de réplicas falhas [Bessani et al. 2014].

Já garantir o item *(ii)* envolve a utilização de um protocolo de difusão atômica [Hadzilacos and Toueg 1994], também conhecido como difusão com ordem total, que possibilita que todas as réplicas corretas entreguem todas as requisições na mesma ordem. Finalmente, para prover o item *(iii)*, é necessário que as operações executadas pelas réplicas sejam deterministas, i.e., que a execução de uma mesma operação (com os mesmos parâmetros) produza a mesma mudança de estado e tenha o mesmo retorno nas diversas réplicas do sistema.

Muito tem se estudado sobre como prover o item *(ii)*, visto que o problema da difusão atômica é equivalente ao do consenso [Hadzilacos and Toueg 1994] (os processos devem entrar em acordo – propriedade fundamental do consenso – acerca da ordem de entrega das mensagens/requisições) sendo geralmente tratado como o gargalo de uma RME por exigir vários passos de comunicação. Em cada passo, uma ou mais réplicas enviam/recebem mensagens para/de uma outra ou mais réplicas. Tanto o número de passos de comunicação quanto a quantidade de mensagens enviadas/recebidas em cada passo variam de acordo com o protocolo utilizado [Castro and Liskov 2002, Abd-El-Malek et al. 2005, Lamport 2006, Marandi et al. 2010, Cowling et al. 2006, Kotla et al. 2009, Guerraoui et al. 2010].



(a) RME Escalável. (b) RME Paralela.

Figura 1. Abordagens para aumentar o desempenho de uma RME.

Um aspecto que recentemente começou a ser explorado é a otimização dos procedimentos necessários para a execução das requisições, modificando a visão sobre a forma

de como o item *(iii)* pode ser atendido. Estas abordagens tiram proveito da semântica das operações para *(1)* dividir o estado da aplicação entre várias RMEs (RME escalável – Figura 1(a)) [Bezerra et al. 2014, Le et al. 2016, da Silva Böge et al. 2016] ou *(2)* executá-las paralelamente nas réplicas (RME Paralela – Figura 1(b)) [Kotla and Dahlin 2004, Marandi et al. 2014, Marandi and Pedone 2014, Alchieri 2015].

RME Escalável: A ideia desta abordagem é dividir o estado da aplicação entre várias RMEs (ou partições) que executam em paralelo (Figura 1(a)). As operações são mapeadas para a partição correspondente e o desempenho é melhorado visto que tanto a ordenação quanto a execução de operações enviadas para RMEs diferentes são executadas em paralelo. No entanto, uma operação que precisa acessar o estado de mais de uma partição deve ser enviada para as várias partições envolvidas e os objetos (estado) devem ser transferidos entre as partições, a fim de que uma delas execute a operação e as outras apenas aguardem esta execução [Bezerra et al. 2014]. Com o objetivo de evitar a execução de operações que envolvem várias partições e a consequente queda de desempenho, o estado da aplicação pode ser redistribuído entre as partições [Le et al. 2016] ou ainda partições podem ser criadas ou removidas [da Silva Böge et al. 2016].

RME Paralela: Em uma RME paralela, cada réplica possui o estado completo da aplicação como em uma RME tradicional, mas operações que acessam partes diferentes do estado (ou partições) são executadas em paralelo através de um conjunto de *threads* de execução (Figura 1(b)). Para isso, duas soluções foram propostas: *(1)* partes do estado da aplicação são associadas a diferentes *threads* e quando uma operação envolve o estado manipulado por várias *threads*, uma delas executa e as outras apenas aguardam [Marandi et al. 2014, Alchieri 2015]; ou *(2)* um grafo de dependências é criado e as *threads* buscam requisições independentes neste grafo para execução [Kotla and Dahlin 2004]. Como podemos perceber, os efeitos negativos causados pela execução de operações que envolvem várias partições tendem a ser menores nesta abordagem visto que não é necessária a transferência de objetos entre partições. Como nas RMEs escaláveis, qualquer técnica utilizada para melhorar o desempenho do protocolo de ordenação pode ser empregada, como a ordenação em lotes [Bessani et al. 2014] ou a execução paralela de várias instâncias do protocolo de ordenação [Marandi et al. 2014].

3. Replicação Máquina de Estados Paralela e Reconfigurável

Esta seção apresenta a nossa proposta para uma RME paralela e reconfigurável. Primeiramente, discutimos o protocolo para execuções paralelas, o qual faz uso de um escalonador (*scheduler*) que atribui as requisições para as *threads* de execução. Posteriormente, apresentamos os protocolos que permitem a reconfiguração do número de *threads* ativas de acordo com políticas de reconfiguração que podem ser definidas pelos usuários.

3.1. Paralelismo

Esta seção apresenta o protocolo para execuções paralelas de requisições em uma réplica, requisito de uma RME paralela (Figura 1(b)). O protocolo aqui apresentado é independente do protocolo de ordenação, o qual também pode executar várias instâncias em paralelo (ordenação em paralelo) [Marandi et al. 2014] e/ou ordenar lotes de requisições em uma única instância (ordenação em lotes) [Bessani et al. 2014].

A ideia deste protocolo é dividir o estado da aplicação entre um conjunto de *threads* de execução, de forma que cada *thread* acesse a sua partição do estado para executar operações em paralelo. Sempre que uma operação envolve o estado de várias *threads*, uma delas acessa todo o estado necessário para executar a operação e as outras *threads* envolvidas aguardam esta execução, garantindo as propriedades de uma RME [Bezerra et al. 2014, Marandi et al. 2014].

Esta divisão do estado define grupos de execução (ou de dependências) e o cliente deve informar para qual grupo de execução sua requisição é endereçada. Em nosso protocolo é possível criar qualquer grupo de execução necessário para a aplicação (basta configurar as réplicas com o identificador do grupo e as *threads* que o compõem), mas os seguintes grupos são definidos por padrão: *(i)* `CONFLICT_ALL` – contém todas as *threads* e deve ser usado para realizar operações que acessam todo o estado da aplicação; *(ii)* `CONFLICT_NONE` – usado para realizar operações que podem executar em paralelo com qualquer outra operação (geralmente operações que não modificam o estado), por qualquer *thread*; *(iii)* cada *thread* ainda possui um grupo com seu próprio identificador (as *threads* são identificadas de forma incremental começando em 0) que deve ser usado para operações que acessam apenas o estado associado à respectiva *thread* (este mapeamento é definido pelo cliente, i.e., operações que acessam uma mesma partição do estado devem sempre ser enviadas para a mesma *thread*).

Algoritmo 1 Algoritmo de distribuição das requisições (*thread scheduler*).

variables: Variables and sets used by the scheduler.
$numThreads \leftarrow$ the number of worker threads
$queues[numThreads] \leftarrow$ the queues used to assign requests to the threads (see Algorithm 2)
$nextThread \leftarrow 0$ // id of the thread that will execute the next `CONFLICT_NONE` request
on initialization:
1) start the barrier for the `CONFLICT_ALL` group with number of parties equal to $numThreads$
2) start the barriers for other created conflict groups
on A-deliver(request):
3) **if** $request.groupId == $ `CONFLICT_NONE` **then**
4) $queues[nextThread].put(request)$ //assigns the request to a thread...
5) $nextThread = (nextThread + 1)\%numThreads$ //...using a roud-robin policy
6) **else if** $request.groupId == $ `CONFLICT_ALL` **then**
7) **for** $i = 0; i < numThreads; i++$ **do** //assigns the request to all threads
8) $queues[i].put(request)$
9) **end for**
10) **else if** $request.groupId < numThreads$ **then** //request directly sent to some thread
11) $queues[request.groupId].put(request)$
12) **else** //request to a created conflict group
13) $group\_queues \leftarrow$ get the queues of the threads in $request.groupId$
14) **for** $each\ q \in group\_queues$ **do** //assigns the request to the threads in the group
15) $q.put(request)$
16) **end for**
17) **end if**

Escalonador. Sempre que uma requisição é entregue pelo protocolo de ordenação, o escalonador é executado (por uma outra *thread*) para atribuir uma determinada requisição a uma ou mais *threads* de execução (Algoritmo 1). A comunicação entre o escalonador e as *threads* de execução ocorre através de uma fila sincronizada, de acordo com o identificador do grupo de execução conforme segue:

- `CONFLICT_NONE` – a requisição é atribuída a uma única *thread* de execução, seguindo a política *round-robin* (linhas 3-5).
- `CONFLICT_ALL` – a requisição é atribuída a todas as *threads* (linhas 6-9).
- Caso a requisição seja endereçada para o grupo de uma *thread* específica, a mesma é atribuída para tal *thread* (linhas 10-12).
- Finalmente, caso a requisição seja endereçada para um grupo formado por mais de uma *thread*, a requisição é encaminhada para as *threads* do grupo (linhas 13-16).

Threads Executoras. Sempre que existir uma requisição disponível para ser executada, cada *thread* procede da seguinte forma, de acordo com o grupo ao qual a requisição foi endereçada (Algoritmo 2):

- `CONFLICT_NONE` ou seu próprio grupo – apenas executa a operação, pois nenhuma sincronização com outras *threads* é necessária (linhas 3-4).
- `CONFLICT_ALL` ou outro grupo – neste caso é necessário que ocorra uma sincronização entre as *threads* do grupo (linhas 5-14), de forma que primeiro é necessário esperar até que todas as *threads* cheguem a este ponto da execução (primeiro acesso à barreira – linhas 7 e 11) para então uma delas executar a requisição (linha 6) enquanto que as outras apenas aguardam por isso (segundo acesso à barreira – linhas 9 e 12). A função $getBarrier(request.groupId).await()$ sinaliza que uma determinada *thread* atingiu a barreira do grupo $request.groupId$.

Algoritmo 2 Algoritmo de execução de requisições (*threads* executoras).

variables: Variables and sets used by each worker thread.
$myId \leftarrow$ id received at initialization // thread id (the ids range from 0 to $maxThreads - 1$)
$queue \leftarrow$ a synchronized/blocking queue that contains the requests to be executed by this thread
on thread run:
1) **while** true **do**
2) $request \leftarrow queue.take()$ //get the next request to be executed, blocks until a request be available
3) **if** $request.groupId ==$ `CONFLICT_NONE` $\vee$ $req.groupId == myId$ **then** // no conflict
4) executes the request against the application state and sends the reply to the client
5) **else** //conflict: $request.groupId ==$ `CONFLICT_ALL` or some other created conflict group
6) **if** $myId == executor\ for\ request.groupId$ **then** // the thread with the smallest id
7) $getBarrier(request.groupId).await()$ // waits the other threads to stop
8) executes the request against the application state and sends the reply to the client
9) $getBarrier(request.groupId).await()$ // resumes the other threads execution
10) **else**
11) $getBarrier(request.groupId).await()$ //signalizes that will wait for the execution
12) $getBarrier(request.groupId).await()$ //waits the execution
13) **end if**
14) **end if**
15) **end while**

3.2. Reconfiguração

Apesar do protocolo da seção anterior implementar uma RME paralela, o número de *threads* de execução, que define o grau de paralelismo, é definido de forma estática na inicialização do sistema. Esta definição é muito importante e afeta diretamente o desempenho do sistema [Marandi et al. 2014, Alchieri 2015]. Um número grande de *threads* tende a aumentar o desempenho em *workloads* com uma grande percentagem de requisições não conflitantes (`CONFLICT_NONE`), mas o desempenho diminui abruptamente com o aumento da quantidade de requisições conflitantes (`CONFLICT_ALL`). Por

outro lado, um número pequeno de *threads* não faz com que o sistema seja capaz de atingir o desempenho máximo para *workloads* com poucas requisições conflitantes. O desempenho máximo do sistema ainda é limitado pelo *hardware* utilizado.

Algoritmo 3 Algoritmo reconfigurável de distribuição das requisições (*thread scheduler*).

variables: Variables and sets used by the scheduler.
$minThreads \leftarrow$ the minimum number of active worker threads
$maxThreads \leftarrow$ the maximum number of active worker threads
$currentThreads \leftarrow$ the current number of active worker threads
$queues[maxThreads] \leftarrow$ the queues used to assign requests to the threads (see Algorithm 4)
$nextThread \leftarrow 0$ // id of the thread that will execute the next `CONFLICT_NONE` request
on initialization:
1) start the barrier for `CONFLICT_ALL` group with number of parties equal to $currentThreads$
2) start the barriers for other created conflict groups
3) start the barrier $reconfig\_barrier$ for `RECONFIG` with number of parties equal to $maxThreads$
on A-deliver(request):
4) $recNum \leftarrow reconfigPolicy(request, minThreads, currentThreads, maxThreads)$
5) **if** $recNum \neq 0 \wedge (minThreads \leq currentThreads + recNum \leq maxThreads)$ **then**
6) $currentThreads \leftarrow currentThreads + recNum$
7) $nextThread \leftarrow 0$
8) $reconfig\_request.groupId \leftarrow$ `RECONFIG`
9) **for** $i = 0; i < maxThreads; i++$ **do** //assigns the request to all threads
10) $queues[i].put(reconfig\_request)$
11) **end for**
12) **end if**
13) **if** $request.groupId ==$ `CONFLICT_NONE` **then**
14) $queues[nextThread].put(request)$ //assigns the request to a thread...
15) $nextThread = (nextThread + 1)\%currentThreads$ //...using a roud-robin policy
16) **else if** $request.groupId ==$ `CONFLICT_ALL` **then**
17) **for** $i = 0; i < currentThreads; i++$ **do** //assigns the request to all active threads
18) $queues[i].put(request)$
19) **end for**
20) **else if** $request.groupId < maxThreads$ **then** //request directly sent to some thread
21) **if** $request.groupId < currentThreads$ **then**
22) $queues[request.groupId].put(request)$
23) **else**
24) $request.groupId \leftarrow$ `CONFLICT_ALL` //handles the request as `CONFLICT_ALL`
25) **for** $i = 0; i < currentThreads; i++$ **do** //assigns the request to all active threads
26) $queues[i].put(request)$
27) **end for**
28) **end if**
29) **else** //request to a created conflict group
30) **if** some thread belonging to $request.groupId$ is not active **then**
31) $request.groupId \leftarrow$ `CONFLICT_ALL` //handles the request as `CONFLICT_ALL`
32) **for** $i = 0; i < currentThreads; i++$ **do** //assigns the request to all active threads
33) $queues[i].put(request)$
34) **end for**
35) **else**
36) $group\_queues \leftarrow$ get the queues of the threads in $request.groupId$
37) **for** $each\ q \in group\_queues$ **do** //assigns the request to the threads in the group
38) $q.put(request)$
39) **end for**
40) **end if**
41) **end if**

Devido ao fato de o *workload* poder mudar durante a execução, e dadas as restrições anteriormente descritas, definir um número ideal para a quantidade de *threads* de execução é uma tarefa bastante difícil, se não impossível, e ao mesmo tempo fundamental para o desempenho do sistema. Neste sentido, esta seção apresenta um protocolo de reconfiguração para uma RME paralela, de forma que o número de *threads* possa sofrer alterações durante a execução da aplicação para tentar se adaptar ao *workload* corrente.

Para isso, este protocolo divide as *threads* em ativas e inativas. Uma *thread* é ativa caso esteja apta a executar requisições, caso contrário é considerada inativa. Somente as *threads* ativas participam dos protocolos da RME paralela e, com isso, afetam o desempenho do sistema. Desta forma, a reconfiguração ocorre através da ativação e desativação de *threads*. Na inicialização do sistema, além de especificar o número inicial de *threads* ativas, o usuário também deve definir os números mínimo e máximo de *threads* que podem estar ativas ao mesmo tempo e fornecer uma política com as regras a serem seguidas para ativação e/ou desativação de *threads*.

Escalonador. Sempre que uma requisição é entregue pelo protocolo de ordenação o escalonador é executado (Algoritmo 3). Primeiramente, a política de reconfiguração é acessada para verificar a necessidade de reconfiguração (linha 4), a qual deve retornar o número de *threads* a serem ativadas (retorno positivo) ou desativadas (retorno negativo). As *threads* ativadas/desativadas sempre serão as de maiores identificadores. Caso seja necessário reconfigurar o sistema (linhas 5-11), uma requisição especial (`RECONFIG`) é adicionada na fila de todas as *threads* (até mesmo as inativas, que também participam da reconfiguração) e o escalonador passa a atribuir requisições para as *threads* ativadas ou a não atribuir mais requisições para as *threads* que serão desativadas. Desta forma, para ativar ou desativar uma *thread* basta começar a adicionar ou parar de adicionar requisições em sua fila, respectivamente. A reconfiguração propriamente dita ocorre quando todas as *threads* atingem o mesmo ponto da execução onde esta requisição especial é executada.

O restante deste protocolo é semelhante ao do escalonador estático (Algoritmo 1). A principal diferença é que apenas as *threads* ativas são consideradas na distribuição das requisições e caso uma requisição seja endereçada para uma *thread* inativa ou para um grupo que a contenha, a mesma é considerada como uma requisição que conflita com todas (`CONFLICT_ALL`) pois de outro modo não seria executada. Requisições endereçadas para grupos que possuem alguma *thread* inativa poderiam ser executadas pelo próprio grupo caso o mesmo fosse reconfigurado (sua barreira reconfigurada para o novo número de *threads* ativas pertencentes ao grupo).

Threads Executoras. A principal diferença para o modelo estático está na execução de reconfigurações (Algoritmo 4), as quais são tratadas de forma semelhante às requisições do grupo `CONFLICT_ALL`, i.e., sem paralelismo. Durante a execução de uma reconfiguração, a barreira do grupo `CONFLICT_ALL` é reconfigurada de acordo com a nova configuração do sistema (linha 6). Todas as *threads* devem participar desta execução pois a barreira utilizada para reconfigurações ($reconfig\_barrier$) deve ser estática (não pode ser reconfigurada em meio a execução da reconfiguração). Como é esperado que a quantidade de reconfigurações seja proporcionalmente muito menor do que a quantidade de requisições de clientes, isso não afeta o desempenho do sistema.

Esta abordagem funciona pelo fato de que até a determinação da reconfiguração pelo escalonador, todas as *threads* ativas na configuração antiga receberam as requisições

e portanto a barreira com a configuração antiga é utilizada até que todas cheguem ao ponto da execução da reconfiguração. A partir da determinação da reconfiguração, o escalonador passa a atribuir requisições apenas para as *threads* ativas na nova configuração e, do ponto da execução da reconfiguração em diante, a barreira estará reconfigurada para refletir este número atual de *threads* ativas.

Algoritmo 4 Algoritmo reconfigurável de execução de requisições (*threads* executoras).

variables: Variables and sets used by each worker thread.
$myId \leftarrow$ id received at initialization // thread id (the ids range from 0 to $maxThreads - 1$)
$queue \leftarrow$ a synchronized/blocking queue that contains the requests to be executed by this thread
on thread run:
1) **while** true **do**
2) $request \leftarrow queue.take()$ //get the next request to be executed, blocks until a request be available
3) **if** $request.groupId ==$ `RECONFIG` **then** // thread reconfiguration
4) **if** $myId == 0$ **then**
5) $reconfig\_barrier.await()$ //waits all the other threads to stop
6) reconfigure the barrier for `CONFLICT_ALL` with number of parties equal to $currentThreads$
7) $reconfig\_barrier.await()$ //resumes all the other threads
8) **else**
9) $reconfig\_barrier.await()$ //signalizes that will wait for the reconfiguration
10) $reconfig\_barrier.await()$ //waits the reconfiguration execution
11) **end if**
12) **end if**
13) lines 3–14 of Algorithm 2
14) **end while**

3.2.1. Políticas de Reconfiguração

A definição da nova configuração do sistema e de quando adotá-la segue uma política de reconfiguração (linha 4 – Algoritmo 3), que pode ser especificada pelo usuário. A política considera a requisição que está sendo escalonada, o número atual de *threads* ativas, além das configurações para o número mínimo e máximo de *threads* ativas.

Algoritmo 5 Política de reconfiguração.

variables: Variables and sets used.
$conflict \leftarrow 0$ // counter for the number of conflict requests
$notConflict \leftarrow 0$ // counter for the number of non conflict requests
$period \leftarrow 10000$ // period (number of requests) to check for reconfigurations
reconfigPolicy(request,minThreads,currentThreads,maxThreads)
1) **if** $request.groupId ==$ `CONFLICT_ALL` **then**
2) $conflict++$
3) **else**
4) $notConflict++$
5) **end if**
6) **if** $(conflict + notConflict) == period$ **then**
7) $percent \leftarrow conflict * 100/period$
8) $conflict \leftarrow 0$; $notConflict \leftarrow 0$
9) **if** $(percent \leq 20) \wedge ((currentThreads + 1) \leq maxThreads)$ **then**
10) **return** 1
11) **else if** $(percent > 20) \wedge ((currentThreads - 1) \geq minThreads)$ **then**
12) **return** -1
13) **end if**
14) **end if**
15) **return** 0

O Algoritmo 5 apresenta um exemplo de política, a qual estabelece que a cada 10.000 requisições é determinado o percentual de requisições conflitantes. Caso o *workload* apresente uma quantidade de até 20% de requisições conflitantes, uma *thread* é ativada até que a quantidade máxima seja atingida. Caso contrário, uma *thread* é desativada até que a quantidade mínima seja atingida. Com um *workload* acima de 20% conflitantes, gasta-se muito tempo sincronizando as *threads* (linhas 5-14 – Algoritmo 2) e geralmente o desempenho é melhor em uma execução sequencial [Marandi et al. 2014, Alchieri 2015].

4. Experimentos

Visando analisar o desempenho das soluções propostas, bem como o comportamento de uma RME com execuções paralelas e reconfigurável, os protocolos propostos foram implementados no BFT-SMART [Bessani et al. 2014] e alguns experimentos foram realizados no Emulab [White et al. 2002]. O BFT-SMART representa a concretização de uma RME, desenvolvida na linguagem de programação Java, onde uma única instância do protocolo de ordenação é executada por vez para ordenar um lote de requisições. O principal objetivo destes experimentos não é determinar os valores máximos de desempenho, mas sim analisar as diferenças entre as abordagens e determinar os ganhos advindos com a possibilidade de reconfiguração.

Aplicação utilizada: Lista Encadeada. Esta aplicação foi implementada nos servidores através de uma lista encadeada (*LinkedList*), que é uma estrutura de dados não sincronizada, i.e., caso duas ou mais *threads* acessem esta estrutura concorrentemente e pelo menos uma delas modifique a sua estrutura, então estes acessos devem ser sincronizados. Neste experimento, a lista foi utilizada para armazenar inteiros (*Integer*) e as seguintes operações foram implementadas para seu acesso: *boolean add (Integer i)* – adiciona i no final da lista e retorna *true* caso i ainda não esteja na lista, retorna *false* caso contrário; *boolean remove(Integer i)* – remove i e returna *true* caso i esteja na lista, retorna *false* caso contrário; *Integer get(int index)* – retorna o elemento da posição indicada; e *boolean contains(Integer i)* – retorna *true* caso i esteja na lista, retorna *false* caso contrário. Note que todas estas operações possuem um custo de $O(n)$, onde n é o tamanho da lista. As seguintes dependências foram definidas para estas operações: add e $remove$ são dependentes de todas as outras operações (`CONFLICT_ALL`), enquanto que get e $contains$ não possuem dependências (`CONFLICT_NONE`), somente as já definidas com add e $remove$.

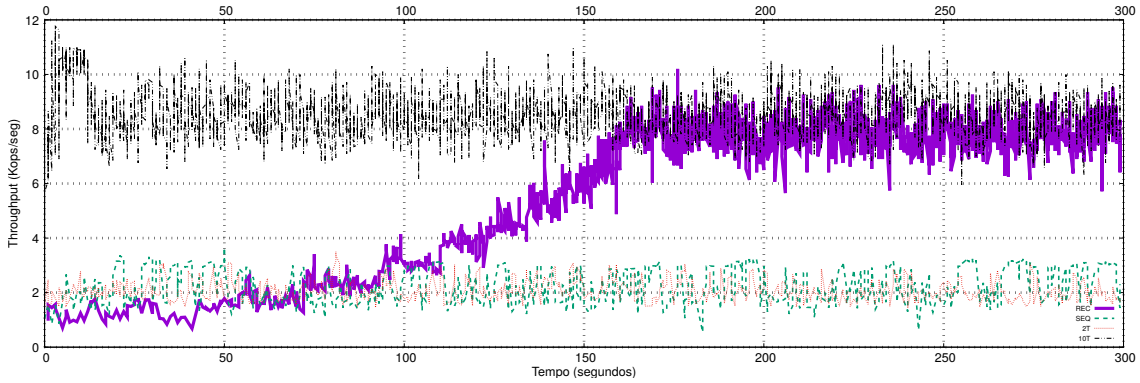
Configuração dos Experimentos. O ambiente para os experimentos foi constituído por 6 máquinas *d430* (2.4 GHz E5-2630v3, com 8 núcleos e 2 *threads* por núcleo, 64GB de RAM e interface de rede gigabit) conectadas a um *switch* de 1Gb. O BFT-SMART foi configurado com 3 servidores para tolerar até uma falha por parada (*crash*). Cada servidor executou em uma máquina separada, enquanto que 90 clientes foram distribuídos uniformemente nas outras 3 máquinas. O ambiente de *software* utilizado foi o sistema operacional Ubuntu 14 64-*bit* e máquina virtual Java de 64 *bits* versão 1.7.0_75.

Nos experimentos, a lista foi inicializada com $100k$ entradas em cada réplica e utilizamos as operações add e $contains$ para execução de operações `CONFLICT_ALL` e `CONFLICT_NONE`, respectivamente. O parâmetro destas operações sempre foi o último elemento da lista ($100k-1$) de forma que em todos os experimentos foi possível executar o mesmo conjunto de operações com os mesmos parâmetros. Para verificar o desempenho das diferentes abordagens, o *throughput* foi medido em um dos servidores (sempre o mesmo – líder do consenso [Bessani et al. 2014]) a cada 1000 requisições durante um

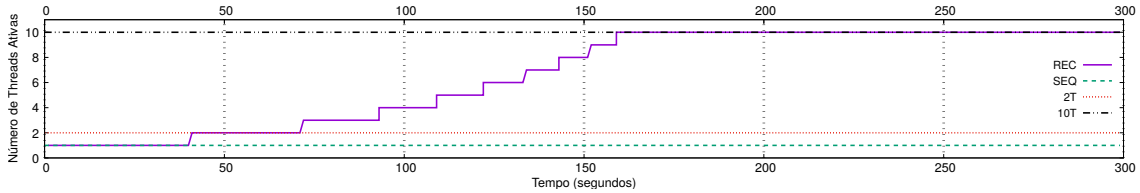
intervalo de 300 segundos. A política de reconfiguração utilizada foi a apresentada no Algoritmo 5, mas utilizamos diferentes períodos para calcular a percentagem de operações conflitantes, conforme descrito nos experimentos.

Resultados e Análises. Três experimentos foram realizados, os quais permitem mostrar que: *(1)* quando o *workload* favorece, o sistema se ajusta para uma configuração com o máximo de *threads* ativas visando aumentar o desempenho; *(2)* quando o *workload* não favorece, o sistema se ajusta para o mínimo de *threads* ativas causando o menor impacto possível no desempenho; e *(3)* quando o workload varia durante a execução, o sistema vai se ajustando para tentar sempre obter o melhor desempenho possível.

No primeiro experimento (Figura 2) o clientes geram um *workload* que favorece o paralelismo (100% `CONFLICT_NONE`) e o sistema foi configurado para iniciar com apenas 1 *thread* ativa, podendo chegar até 10. O período para verificar a necessidade de reconfigurações foi especificado para cada 50.000 requisições. Para efeitos de comparação, os gráficos mostram os valores do *throughput* para o sistema com execuções paralelas e reconfigurações (REC), com execução sequencial que representa uma RME tradicional (SEQ), sem reconfigurações mas paralelo configurado com 2 (2T) e 10 (10T) *threads*. Apesar da melhor configuração para este *workload* ser a com 10 threads, isso só seria possível com um conhecimento prévio a respeito das operações que seriam executadas no sistema (neste caso apenas operações $contains$), o que não é realista.



(a) *Throughput*.

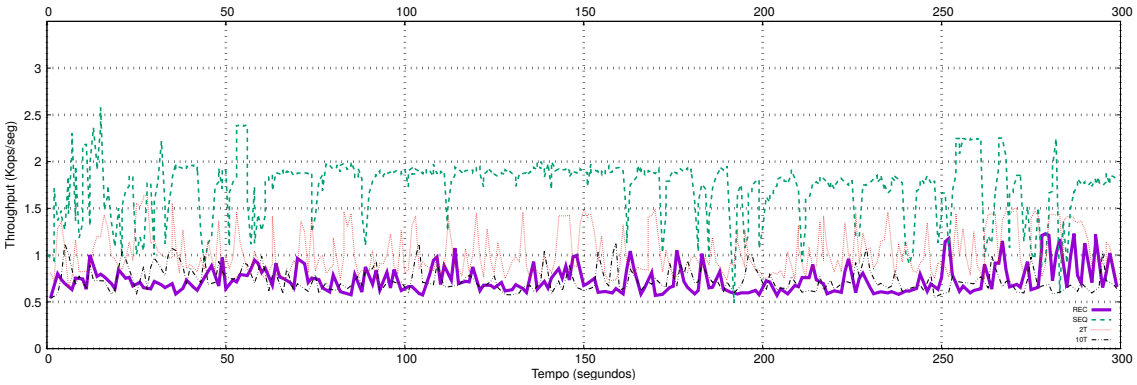


(b) Número de *threads* ativas.

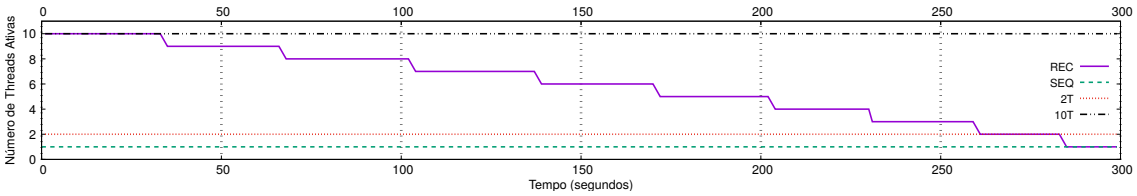
Figura 2. *Throughput* para *workload* de 0% conflitantes.

É possível perceber que com o passar da execução, o sistema com reconfiguração vai ativando as *threads* até atingir o limite máximo de 10, fazendo com que o desempenho seja praticamente o mesmo da configuração 10T. Outro ponto que merece destaque é que com apenas uma *thread* ativa, o desempenho é suavemente inferior ao da execução sequencial devido aos gastos na comunicação entre o escalonador e a *thread* através da fila sincronizada.

O segundo experimento (Figura 3) representa um cenário oposto ao primeiro e mostra que quando o *workload* não favorece (100% `CONFLICT_ALL`), mesmo inicializado com uma configuração inadequada (10 *threads* ativas), o sistema vai desativando *threads* até atingir o limite mínimo de 1 *thread* ativa causando o menor impacto possível no desempenho. Neste experimento, o período para verificar a necessidade de reconfigurações foi configurado para 25.000 requisições. Novamente são apresentados os valores para o sistema configurado como paralelo e reconfigurável (REC), sequencial (SEQ), paralelo com 2 (2T) ou 10 (10T) *threads*. Vale destacar que conforme as *threads* vão sendo desativadas através de reconfigurações, recursos podem ir sendo desalocados.



(a) *Throughput*.



(b) Número de *threads* ativas.

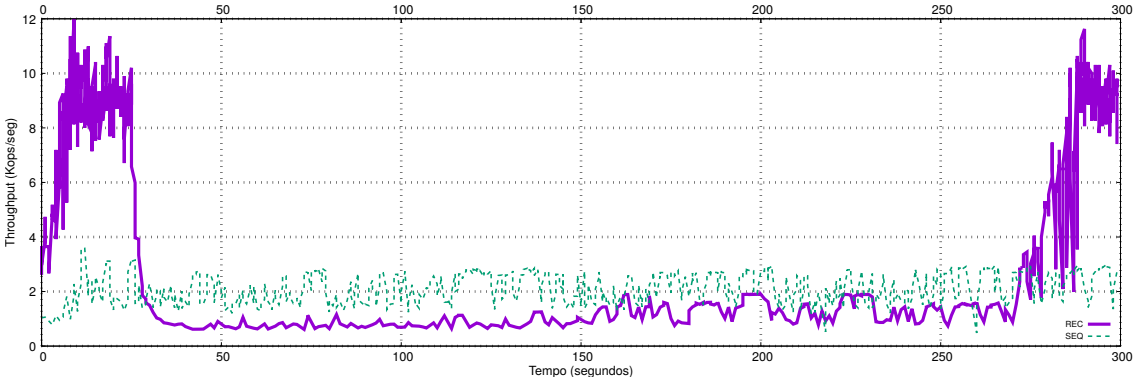
Figura 3. *Throughput* para *workload* de 100% conflitantes.

Finalmente, o terceiro experimento (Figura 4) representa um cenário em que o *workload* sofre variações durante a execução: inicialmente cada cliente executa 5.000 requisições não conflitantes, depois passam a executar 5.000 requisições conflitantes e no final voltam a executar operações não conflitantes. Na execução com reconfiguração, o sistema foi inicializado com 5 *threads* ativas e o período de reconfigurações foi definido como a cada 10.000 requisições.

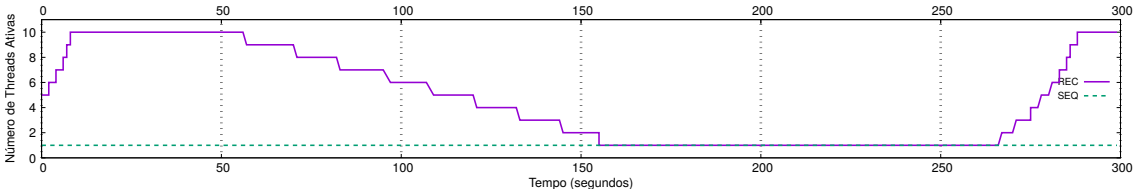
Através deste experimento podemos perceber que o sistema inicialmente ativa *threads* até atingir o limite máximo para obter o melhor desempenho visto que o *workload* favorece. Quando os clientes passam a executar operações conflitantes o sistema passa a desativar as *threads*, as quais são ativadas novamente quando os clientes voltam a executar operações não conflitantes.

Nestes experimentos utilizamos uma política de reconfiguração mais conservadora que ativa ou desativa apenas uma *thread* a cada reconfiguração, o que permitiu mostrar o comportamento dos protocolos propostos. Porém, políticas mais agressivas podem ser utilizadas com o objetivo de se atingir a configuração ideal mais rapidamente. Além disso, analisando sobre outra perspectiva, a taxa de conflitos é algo externo aos servidores e independe do estado interno de alocação e utilização de recursos. Neste sentido, uma

política poderia criar níveis intermediários de paralelismo buscando mapear intervalos de taxas de conflitos para um número de *threads* ativas de forma que o balanço entre desempenho e ociosidade de recursos seja ótimo.



(a) *Throughput*.



(b) Número de *threads* ativas.

Figura 4. *Throughput* variando o *workload* durante a execução.

5. Conclusões

Neste trabalho apresentamos protocolos para implementação de uma RME paralela e reconfigurável, permitindo o ajuste do sistema para o *workload* atual. Não é necessário o conhecimento de nenhuma informação adicional antes da execução do sistema e experimentos mostram os ganhos advindos com a possibilidade de reconfiguração.

Como trabalhos futuros, pretendemos explorar outras formas de escalonar as requisições, como o escalonamento em lotes de forma que o custo da sincronização entre as *threads* seja diluído entre as requisições do lote e, principalmente, analisar políticas de reconfiguração baseadas em outros parâmetros, como o *throughput* e/ou a latência atual apresentada pelo sistema, além do cenário atual de alocação de recursos.

Referências

Abd-El-Malek, M., Ganger, G., Goodson, G., Reiter, M., and Wylie, J. (2005). Fault-scalable Byzantine fault-tolerant services. In *ACM Symposium on Operating Systems Principles*.

Alchieri, E. A. P. (2015). Suportando execuções paralelas no BFT-SMaRt. In *Anais do XVI Workshop de Teste e Tolerância a Falhas- WTF 2015*.

Bessani, A., Sousa, J., and Alchieri, E. (2014). State machine replication for the masses with BFT-SMaRt. In *International Conference on Dependable Systems and Networks*.

Bezerra, C. E., Pedone, F., and Renesse, R. V. (2014). Scalable state-machine replication. In *44th IEEE/IFIP International Conference on Dependable Systems and Networks*.

Castro, M. and Liskov, B. (2002). Practical Byzantine fault-tolerance and proactive recovery. *ACM Transactions on Computer Systems*, 20(4):398–461.

Cowling, J., Myers, D., Liskov, B., Rodrigues, R., and Shrira, L. (2006). HQ-Replication: A hybrid quorum protocol for Byzantine fault tolerance. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*.

da Silva Böge, D., da Silva Fraga, J., and Alchieri, E. (2016). Reconfigurable scalable state machine replication. In *Proceedings of the 2016 Latin-American Symposium on Dependable Computing*.

Guerraoui, R., Knežević, N., Quéma, V., and Vukolić, M. (2010). The next 700 BFT protocols. In *Proceedings of the ACM SIGOPS/EuroSys European Systems Conference*.

Hadzilacos, V. and Toueg, S. (1994). A modular approach to the specification and implementation of fault-tolerant broadcasts. Technical report, Department of Computer Science, Cornell.

Herlihy, M. and Wing, J. M. (1990). Linearizability: A correctness condition for concurrent objects. *ACM Trans. on Programing Languages and Systems*, 12(3):463–492.

Kotla, R., Alvisi, L., Dahlin, M., Clement, A., and Wong, E. (2009). Zyzzyva: Speculative Byzantine fault tolerance. *ACM Transactions on Computer Systems*, 27(4):7:1–7:39.

Kotla, R. and Dahlin, M. (2004). High throughput byzantine fault tolerance. In *Proceedings of the International Conference on Dependable Systems and Networks*.

Lamport, L. (1998). The part-time parliament. *ACM Transactions Computer Systems*, 16(2):133–169.

Lamport, L. (2006). Fast paxos. *Distributed Computing*, 19(2):79–103.

Le, L. H., Bezerra, C. E., and Pedone, F. (2016). Dynamic scalable state machine replication. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 13–24.

Marandi, P. J., Bezerra, C. E., and Pedone, F. (2014). Rethinking state-machine replication for parallelism. In *Proc. of the 34th Int. Conference on Distributed Computing Systems*.

Marandi, P. J. and Pedone, F. (2014). Optimistic parallel state-machine replication. In *Proceedings of the 33rd International Symposium on Reliable Distributed Systems*.

Marandi, P. J., Primi, M., Schiper, N., and Pedone, F. (2010). Ring paxos: A high-throughput atomic broadcast protocol. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks*.

Schneider, F. B. (1990). Implementing fault-tolerant service using the state machine aproach: A tutorial. *ACM Computing Surveys*, 22(4):299–319.

White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., and Joglekar, A. (2002). An Integrated Experimental Environment for Distributed Systems and Networks. In *Proc. of 5th Symp. on Operating Systems Design and Implementations*.

Zbierski, M. (2015). Parallel byzantine fault tolerance. In Wilinski, A., Fray, I. E., and Pejas, J., editors, *Soft Computing in Computer and Information Science*, volume 342 of *Advances in Intelligent Systems and Computing*, pages 321–333. Springer Publishing.

Salão de Ferramentas do SBRC 2017
Sessão Técnica 1
Rede sem Fio e IoT I

BWPING-UDP – Avaliando o Desempenho de Redes Sem Fio

Henrique Duarte Moura[1], **Erik de Britto e Silva**[12], **Daniel F. Macedo**[1]

[1] Departamento de Ciência da Computação – Universidade Federal de Minas Gerais

[2]Departamento de Computação e Sistemas
Universidade Federal de Ouro Preto (UFOP) – Campus João Monlevade

henriquemoura@dcc.ufmg.br, erik@dcc.ufmg.br, damacedo@dcc.ufmg.br

Abstract. *Performance evaluation is an important process on the development of new protocols and networks. The existing tools are not suited to evaluate wireless networks, since they neglect features such as packet losses and jitter caused by retransmissions at the MAC layer, which are commonplace in wireless networks. This paper presents bwping-udp, a performance evaluation tool for wireless networks that measures throughput, jitter and loss rate. Using examples, we show how bwping-udp can be used to evaluate wireless and Ethernet networks.*

Resumo. *A avaliação de desempenho é um processo muito importante no desenvolvimento de novos protocolos e redes. Entretanto, as ferramentas de avaliação de desempenho existentes não são adaptadas para a avaliação de redes sem fio, negligenciando aspectos como a taxa de perda dos enlaces e o alto jitter, decorrente das retransmissões frequentes na sub-camada MAC em redes sem fio. Este artigo apresenta o bwping-udp, uma ferramenta para avaliação de desempenho de redes sem fio que mede a vazão, o atraso, o jitter e a taxa de perda de pacotes. A partir de exemplos, mostramos como bwping-udp pode ser usado para a avaliação de redes sem fio e redes Ethernet.*

1. Introdução

A medição do desempenho entre hospedeiros em uma rede de computadores, como a Internet, e, em especial, em uma rede sem fio, é fundamental para diagnosticar problemas de desempenho atuais, bem como para projetar futuros serviços distribuídos. Infelizmente, a arquitetura da Internet não foi projetada com um objetivo primário de medição de desempenho [Clark 1988] e, portanto, os protocolos da rede possuem poucos serviços "embutidos" que suportam essa necessidade. Portanto, são necessários instrumentos de medição adicionais capazes de prover estes serviços de medição, a serem fornecidos separadamente, ou devem ser implementadas novas funcionalidades na pilha de protocolos voltadas para a medição.

Neste artigo, argumentamos que o comportamento do UDP (*User Datagram Protocol*) comumente implantado na camada de transporte da Internet pode ser usado como um serviço de medição. Apresentamos uma nova ferramenta, chamada *bwping-udp*, que utiliza *sockets* UDP para medir as taxas de perda de pacotes, atraso e variação do atraso entre um hospedeiro de origem e um hospedeiro de destino. Ao contrário das tradicionais ferramentas de medição de desempenho, *bwping-udp* é capaz de distinguir quais perdas ocorrem na direção origem-destino e quais ocorrem no sentido inverso. Por não utilizar TCP (*Transmission Control Protocol*), como no caso do *iperf3*, *bwping-udp* é capaz

de melhor explorar as limitações de largura de banda da rede, não estando sujeito aos mecanismos de controle de fluxo e controle de congestionamento. *bwping-udp* é uma ferramenta portável e simples, pois não necessita outros recursos além da biblioteca de *sockets* do Unix.

A contribuições deste trabalho são uma ferramenta mais completa para medição de desempenho em redes tanto *Ethernet* quanto sem fio, bem como uma comparação entre as diversas ferramentas disponíveis. A ferramenta *bwping-udp* já foi empregada em artigos na área de redes sem fio e *Ethernet* produzidos na UFMG, como: (a) Avaliação de desempenho de Redes IEEE 802.11p [Teixeira et al. 2014]; (b) Ethanol [Moura et al. 2015]; e (c) Avaliação de desempenho de switches OpenFlow [Costa et al. 2016].

O restante do artigo está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 apresenta o *bwping-udp*. Exemplos de utilização desta ferramenta são mostrados na Seção 4. Por fim, na Seção 5 apresentamos as conclusões deste trabalho.

2. Trabalhos Relacionados

Existem diversas ferramentas que utilizam recursos dos protocolos TCP, UDP ou ICMP (*Internet Control Message Protocol*) para realizar medições de desempenho da rede. Apresentamos na Tabela 1 uma comparação de características das diversas ferramentas apresentadas nesta seção. Um ✓indica que a característica está presente. Se a ferramenta é implementada como cliente-servidor, esta característica é indicada na coluna *CS*. A ferramenta pode utilizar streams TCP para realizar a medição, indicado na coluna *TCP*, ou usar utiliza fluxos UDP para realizar as medições - coluna *UDP*. A ferramenta pode utilizar diferentes métricas: (1) a vazão no enlace (*Vazão*); (2) o atraso de propagação (*Atraso*); (3) a variação do atraso de propagação (*Jitter*); e (4) a perda de pacotes (*Perdas*). A indicação se ferramenta calcula o desvio padrão das medidas é feita na coluna *SD*. A coluna *IP* indica se há suporte a IP versão 4 (v4), versão 6 (v6) ou ambos. Está indicado na coluna *W* se a ferramenta permite definir o tamanho da janela TCP ou o tamanho do *payload* em UDP. A coluna *PF* mostra se a ferramenta permite criar diversos fluxos em paralelo em um só processo. A coluna *MC* indica se a ferramenta oferece suporte à comunicação multicast.

Ferramentas baseadas em ICMP (*Internet Control Message Protocol*), como *ping*, *fping* e *traceroute*, enviam pacotes de teste para um hospedeiro e medem a perda observando se os pacotes de resposta chegam ou não dentro de um período de tempo. Há dois problemas principais com esta abordagem: (1) *Assimetria das perdas* e (2) *Filtragem ICMP*. A taxa de perda de pacotes no caminho para frente pode ser bastante diferente da taxa de perda de pacotes no caminho inverso entre dois hospedeiros. Sem qualquer informação adicional do hospedeiro de destino, é impossível para uma ferramenta baseada em ICMP determinar se o pacote de sonda foi perdido ou se a resposta foi perdida. Ferramentas baseadas em ICMP dependem da implementação dos serviços *ICMP Echo* e *ICMP Time Exceeded* para obter a resposta de um hospedeiro, contudo no percurso do pacote sonda podem existir mecanismos que restringem a eficácia destas ferramentas. Por exemplo, o sistema operacional de um hospedeiro permite limitar a taxa de respostas ICMP, aumentando, portanto, a taxa de perda de pacotes relatada por *ping*. Abordagens mais drásticas dos operadores de rede podem levar à filtragem completa dos pacotes ICMP. Podemos, ainda, encontrar situações de *spoofing* de ICMP, quando um nó da

Tabela 1. Comparação entre ferramentas de medição de desempenho em redes

Ferramenta	CS	TCP	UDP	ICMP	Vazão	Atraso	Jitter	Perda	PF	SD	W	IP	MC	Versão
ping/ping6				✓		✓		✓			✓*4	v4, v6		Iputils
traceroute traceroute6				✓		✓						v4, v6		s20121221
fping/fping6 [I]				✓		✓		✓	✓		✓*4	v4, v6		3.0
thrulay-hd [II]	✓	✓	✓			✓			✓		✓	v4, v6		0.6.4
iperf [III]	✓	✓*	✓		✓	✓*3	✓*3	✓			✓	v4, v6	✓	3.0
httperf [IV]		✓*2			✓			✓	✓		✓	v4, v6		0.9.1
netperf [V]	✓	✓	✓		✓						✓	v4, v6		2.5.0
ttcp [VI]	✓	✓	✓		✓						✓	v4		
nttcp [VI]	✓	✓	✓		✓						✓	v4	✓	
nuttcp [VI]	✓	✓	✓		✓			✓*3				v4, v6		8.1.4
bwping-udp	✓		✓		✓	✓	✓	✓		✓	✓	v4, v6		1.0

Notas:
* Implementa ainda SCTP
*2 Necessita de um servidor HTTP (como hospedeiro de destino) para funcionar
*3 Somente para as medições usando UDP
*4 Permite indicar o número de bytes enviados no pacote ICMP

Referências:
[I] http://fping.org/
[II] http://thrulay-hd.sourceforge.net/
[III] https://iperf.fr/
[IV] https://github.com/httperf/httperf
[V] http://www.netperf.org/netperf/
[VI] https://www.nuttcp.net/

rede responde às solicitações ICMP em nome do hospedeiro de destino, impedindo assim medições de ponta a ponta reais. Estas situações são, na sua maioria, contornadas pela implementação do *bwping-udp*, pois é uma ferramenta instalada nas duas extremidades do caminho a ser avaliado. Contudo nossa ferramenta não consegue resolver as situações de *traffic shapping*, *rate limiting* ou de filtragem de tráfego UDP.

As ferramentas netperf, nuttcp, ttcp e nttcp permitem medições tanto em TCP quanto em UDP, contudo a métrica medida é a vazão, sendo que as duas últimas somente funcionam em IP versão 4. nttcp obtem ainda a perda em medições UDP. O thrulay-hd funciona também em TCP e UDP e obtem o atraso em IP versão 4 e 6. O *bwping-udp* apesar de funcionar somente com UDP, oferece mais métricas que estas ferramentas. O httperf, ferramenta utilizada para avaliação de desempenho de servidores HTTP, ocupa alta porcentagem da CPU, acima de 90% conforme é citado em sua documentação, e retorna as medições de tempo com precisão de milissegundos, ao passo que a *bwping-udp* não necessita de tanto processamento quando em execução e retorna os tempos com precisão de microssegundos. Embora httperf apresente muitas informações e um grande número de variações de carga, tanto em rajadas de requisições paralelas ou não, utiliza conexões TCP, ficando sujeito ao controle de fluxo e de congestionamento. *bwping-udp*, ao utilizar UDP, possui menores atrasos, por não estabelecer uma conexão. Ele avalia diretamente o menor tempo de RTT (*Round Trip Time*) entre a origem e o destino.

3. A Ferramenta bwping-UDP

O *bwping-udp* é uma ferramenta para avaliação de desempenho de redes que mede a vazão, o atraso, o $jitter$ e a taxa de perda de pacotes. O software ainda obtém os valores de sinal e ruído da interface sem fio, bem como o percentual de uso e de ociosidade da CPU. Estes dados são importantes para redes sem fio, por exemplo para a determinação da distância máxima de comunicação de um enlace ou para ajustar a distância entre repetidores em uma instalação de longa distância. O *bwping-udp* é um software livre. O código fonte e a sua documentação estão disponíveis no endereço https://github.com/h3dema/bwping-udp. Um vídeo mostrando o *download*, a instalação e o uso do *bwping-udp* pode ser encontrado em https://youtu.be/ViU3VMvwVRo.

A ferramenta possui mecanismos que facilitam o desenvolvimento de trabalhos científicos. Por exemplo, os relatórios das métricas coletadas apresentam a sua média e variância. Outro aspecto importante é que os resultados são sempre associados a um *timestamp*, para que possam ser correlacionados com a saída de outras ferramentas. Iremos apresentar a seguir um exemplo de medição de desempenho de redes veiculares, onde os dados de desempenho foram associados com medidas de GPS. Finalmente, os dados podem ser exportados em formato CSV, de forma que possam ser mais facilmente tratados por programas de análise de dados. A ferramenta pode ser empregada para:

- **Geração de carga para experimentos em redes reais.** O *bwping-udp* pode gerar carga em taxas fixas. Tal característica permitiria avaliar, por exemplo, como o algoritmo de contenção se comporta ao variarmos a quantidade de clientes com uma carga fixa, ou como ele se comporta quando variamos a carga dos clientes.
- **Medição do desempenho de pico de enlaces sem fio.** Ao colocarmos uma taxa de envio mais alta que o enlace suporta, podemos avaliar a sua vazão máxima. Como nas redes sem fio as perdas devido a erros de transmissão e a colisões são muito frequentes, é importante que esta medição seja confrontada com a vazão do enlace. Isto permitiria identificar, por exemplo, situações onde estamos avaliando o enlace de uma posição desfavorável (obstáculos entre o transmissor e receptor ou com uma grande distância entre eles).
- **Realização de** *site surveys*. O desempenho de uma rede sem fio depende da distância entre o emissor e o receptor, bem como a existência de obstáculos. Assim, redes sem fio de grande porte requerem uma avaliação do desempenho da mesma em diversos pontos (*site survey*). Estas medições são úteis tanto para identificar o desempenho da rede em diferentes pontos do ambiente, quanto para auxiliar na identificação de pontos de baixa cobertura e desempenho, que são fortes candidatos para a instalação de mais pontos de acesso ou repetidores.

O *bwping-udp* opera da seguinte forma. Existe um executável para o cliente, que envia os dados, e outro para o servidor. O cliente e servidor devem concordar em uma porta a ser empregada na medição. O usuário pode definir parâmetros de execução tais como o tamanho do *payload* (uma vez que a vazão e a taxa de erro em redes sem fio dependem do tamanho do *payload*), o intervalo entre geração de pacotes (para geração de carga em experimentos) e o intervalo entre a impressão de relatórios.

O *bwping-udp* foi desenvolvido na linguagem C. Assim, a ferramenta possui baixo consumo de CPU e memória, e requer somente bibliotecas comuns a qualquer ambiente Unix. Compilamos e testamos seu uso nas plataformas Intel/AMD utilizando diversos sabores do Linux 32/64-bits, Raspberry Pi 2 e Pi 3 B usando o sistema Raspbian Jesse[1] e em roteadores domésticos de baixo custo Linksys WRT54G[2] e TP-Link com OpenWrt[3]. A ferramenta suporta tanto redes IPv4 quanto redes IPv6.

4. Exemplos de experimentos

Nesta seção apresentamos alguns exemplos retirados de artigos publicados que utilizaram a ferramenta *bwping-udp*. Agradecemos aos autores dos artigos [Teixeira et al. 2014, Moura et al. 2015, Costa et al. 2016] pela gentil cessão de seus resultados para compor as seções 4.1, 4.2 e 4.3.

[1]https://www.raspberrypi.org/downloads/raspbian/

[2]http://www.linksys.com/br/p/P-WRT54GL/

[3]https://wiki.openwrt.org/

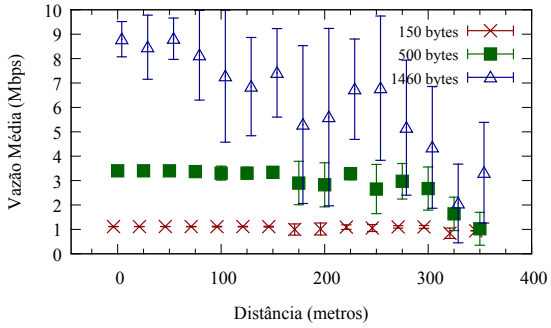


Figura 1. Vazão média para a velocidade de 40 km/h.

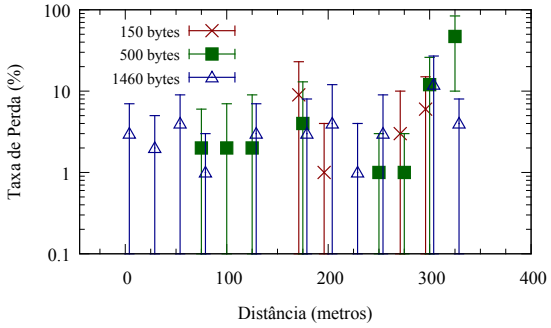


Figura 2. Taxa de perda para a velocidade de 60 km/h.

4.1. Rede veicular

No artigo [Teixeira et al. 2014] os autores empregaram o *bwping-udp* para avaliar o desempenho de enlaces IEEE 802.11p. Foram empregados dois notebooks com placas IEEE 802.11p. Estes foram colocados em carros, sendo um estacionário e o outro em movimento. Um dos objetivos do artigo foi avaliar a distância máxima de comunicação deste padrão, bem como o seu desempenho, em termos de largura de banda e perda de pacotes. Para tanto, foram instalados receptores GPS nos notebooks, para identificar a posição dos carros em cada momento. Os dados do GPS foram registrados em arquivo e cruzados *offline* com os dados do *bwping-udp*, para que o desempenho obtido fosse geo-localizado.

Os gráficos 1 e 2 apresentam a vazão e a taxa de perda, respectivamente, para veículos se movendo a diferentes velocidades. O eixo X indica a distância entre os veículos e o eixo Y foi plotado em escala logarítmica. Vemos que o desempenho da rede é afetado pelo tamanho do pacote transmitido: pacotes maiores permitem uma maior vazão. As medições de taxa de erro ainda permitiram identificar que a qualidade do enlace é degradada com o aumento da velocidade dos veículos.

4.2. Medição de vazão em redes sem fio definidas por software

O Ethanol [Moura et al. 2015] implementa o conceito de redes sem fio definidas por software, onde um controlador possui visão global da rede e é capaz de gerenciá-la. Para demonstrar esta capacidade, os autores apresentam um experimento que simula uma rede doméstica ou empresarial. Esta rede é formada por um roteador R que possui uma interface sem fio e quatro interfaces Ethernet. A este roteador estão conectados um servidor S via rede ethernet e três clientes: C_1 é uma estação sem fio e C_2 e C_3 são duas estações com conexão Ethernet. O computador S tem a ferramenta *bwping-server* instalada, enquanto os clientes possuem o cliente do *bwping-udp*. Na primeira fase do experimento é medida a vazão do tráfego quando C_1 e C_2 demandam simultaneamente recursos da rede. Já na segunda fase, a partir de $170s$, o cliente C_3 é ativado e começa a demandar recursos da rede, como podemos ver pelo aparecimento da linha azul (marcada com um circulo cheio azul) na figura 4.

Para mostrar a capacidade do controlador em gerenciar o fluxo nas redes sem fio e Ethernet, foram feitos dois experimentos: um sem o controlador e outro com o controlador dando prioridade ao tráfego da rede sem fio. Na figura 3 mostramos os resultados obtidos em [Moura et al. 2015] para as medições realizadas com o *bwping-udp* para a rede sem o uso do controlador, onde observamos que o cliente sem fio quase não consegue enviar

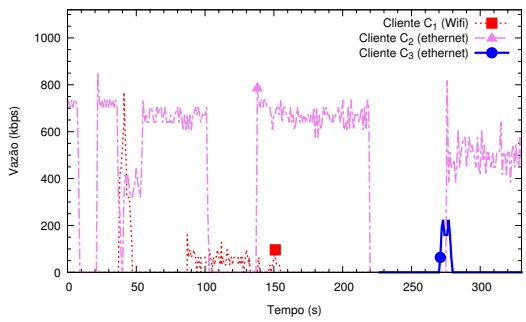


Figura 3. Vazão média em kbps na rede sem fio sem o controlador Ethanol

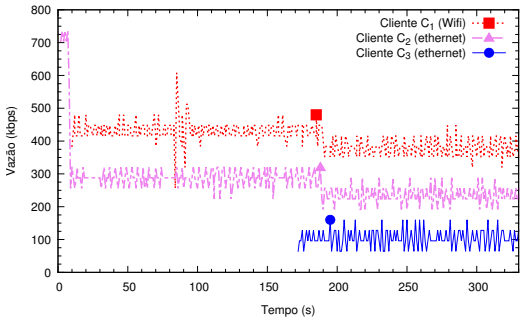


Figura 4. Vazão média em kbps na rede sem fio com o controlador Ethanol

dados ao computador S. Enquanto na figura 4 são mostrados os resultados obtidos com o uso do controlador gerenciando os fluxos, quando o cliente sem fio consegue maior acesso à rede.

4.3. Medição de desempenho em switches

O desempenho de vários switches podem ser comparados entre si quanto as métricas de latência, de vazão e de jitter entre dois hospedeiros que trocam pacotes entre si. Usando um testbed montado no laboratório WINET do DCC/UFMG, vários switches OpenFlow distintos sofreram medição destas métricas através da ferramenta *bwping-udp*. Foi montada uma disposição padrão de um controlador POX e dois hospedeiros que utilizaram o *bwping-udp* conectados a cada switch OpenFlow. As comparações foram feitas utilizado curvas de função de distribuição cumulativa (CDF). Na figura 5, mostramos os resultados obtidos em [Costa et al. 2016]. As medições da latência de nove switches OpenFlow realizadas com o *bwping-udp* utilizando pacotes com payload de 256 bytes. Na figura 6, os resultados das medições de jitter são mostrados. Neste caso, o *bwping-udp* foi configurado para utilizar pacotes com payload de 64 bytes.

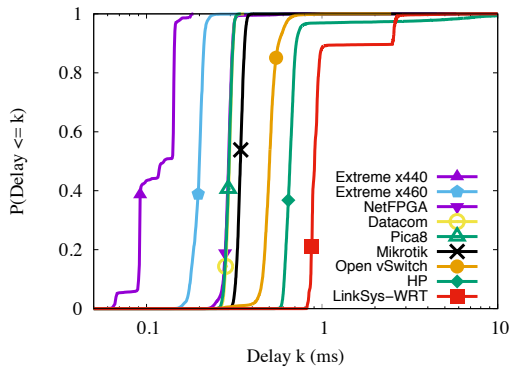


Figura 5. Comparação da latência de Switches OpenFlow

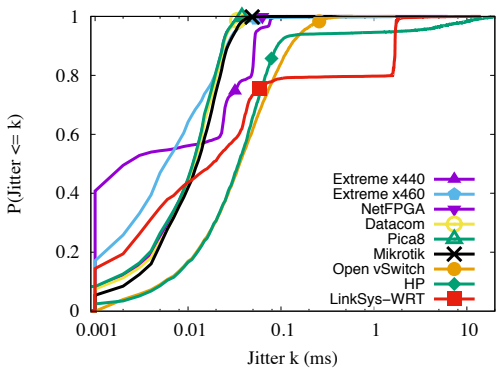


Figura 6. Comparação do jitter de Switches OPenFlow

4.4. Rede virtual privada - VPN

A ferramenta *bwping-udp* pode ser utilizada para analisar o comportamento da rede entre dois computadores interligados via VPN (*Virtual private network*). Este experimento

utiliza dois computadores conectados via ethernet à sua rede local. Cada uma das redes locais possui acesso à Internet. O computador C_1 com o *bwping-server* foi instalado no laboratório WINET e, portanto, utiliza a rede de universidade para acesso à internet. O roteador VPN (R_1) está localizado no DCC, assim de C_1 ao roteador $R1$ são 3 saltos, identificados utilizando *traceroute*. O outro computador C_2 com o *bwping-client* foi instalado na residência de um dos autores. A rede local de C_2 possui acesso à Internet via um provedor de banda larga brasileiro, sendo C_2 conectado via Ethernet diretamente ao roteador internet. O computador C_2 estabelece uma VPN com o roteador R_1, desta forma C_2 tem acesso a C_1. A conexão VPN utiliza o protocolo PPTP (*Point-to-Point Tunneling Protocol*). A MTU (*Maximum Transmission Unit*) para a VPN é de 1400 bytes, inferior à MTU da rede ethernet dos dois computadores, que é de 1500 bytes.

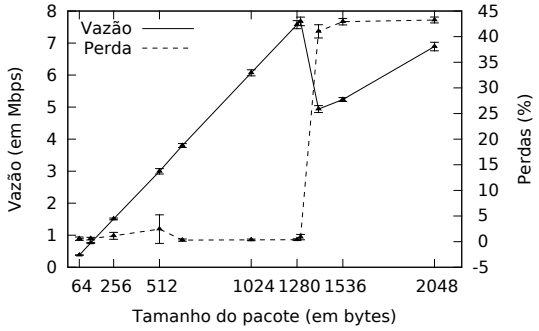


Figura 7. Vazão e perdas para conexão VPN

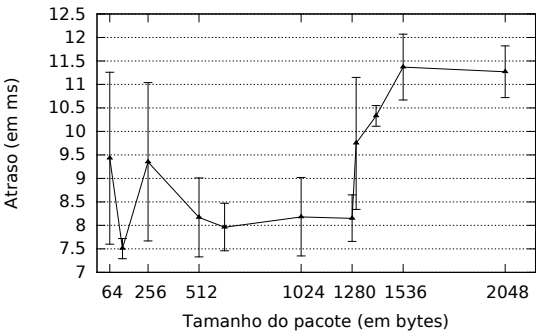


Figura 8. Atraso na VPN

Foram coletadas 10.000 amostras com pacotes de 64 até 2048 bytes de *payload*. Os resultados são mostrados nas figuras 7 e 8. A figura 7 mostra a vazão medida em C_2. Observamos uma queda de vazão para pacotes com *payload* entre 1300 e 1400. Neste intervalo está o valor da MTU da VPN. A figura 7 mostra ainda que a partir deste valor, a quantidade de perdas (em percentual do total de pacotes transmitidos) aumenta, explicando a queda na vazão. Observamos uma recuperação na vazão à medida que o *payload* aumenta depois deste ponto, mesmo com o aumento da perda. Isto ocorre pois os fragmentos que não são perdidos têm tamanhos maiores. A figura 8 mostra os atrasos, em milissegundos, medidos na conexão VPN. A conexão VPN influencia muito esta métrica, como podemos identificar pelos intervalos de confiança.

4.5. Rede Ad-Hoc

A ferramenta *bwping-udp* pode ser utilizada para gerar os dados necessários para montagem um mapa de calor em um ambiente residencial para os parâmetros medidos, como mostrado nas figuras 9 e 10. Para demonstrar esta capacidade, utilizamos dois computadores com placas de rede sem fio. O computador C_1 com o *bwping-server* foi ligado em um dos cômodos da casa. A localização deste computador é identificada nas figuras 9 e 10 com os dizeres "bwping-server". O outro computador - C_2 - com o *bwping-client* foi movimentado entre os cômodos. Foram selecionados 64 pontos para amostragem em um grid regular com espaçamento de $1,1m$ na horizontal e $0,9m$ na vertical. Em cada um destes pontos foram coletadas 5.000 amostras com pacotes com 64, 128, 256, 512, 1024 e 2048 bytes de *payload*. Mostramos somente os resultados para *payload* de 1024 bytes.

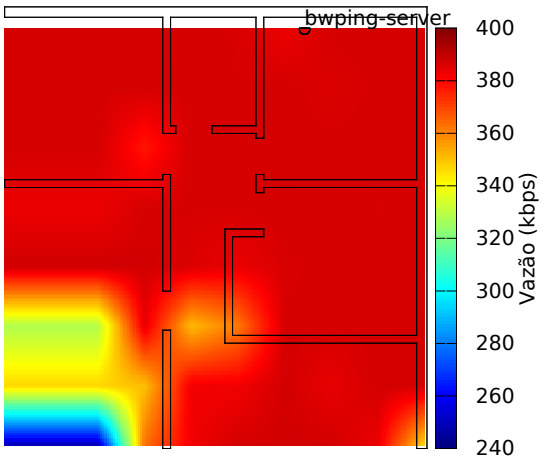


Figura 9. Vazão em kbps para rede ad-hoc

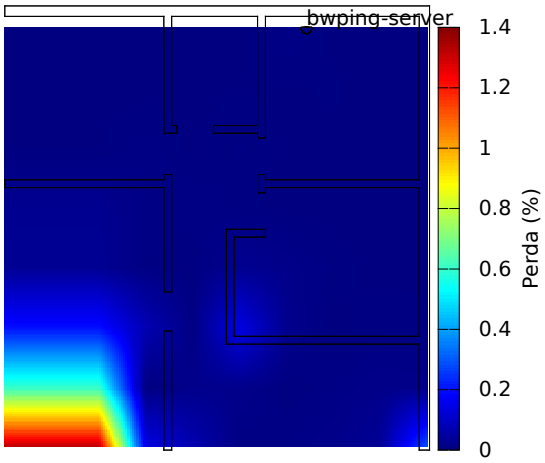


Figura 10. Percentual de perdas para rede ad-hoc

Identificamos que a vazão não varia muito nos cômodos próximos a C_1, como mostrado pela grande mancha em vermelho na figura 9. Somente quando a distância começa a ficar maior que $15m$ ou no caso da existência de pelo menos 4 paredes de alvenaria entre os dois computadores é que a vazão começa a reduzir. Uma quantidade maior de perdas é detectada na mesma área onde a vazão sofre um grande impacto. Este local é identificado pela mancha vermelha na figura 10.

5. Conclusões

Este artigo apresentou o *bwping-udp*, que é um software para medição de desempenho em redes sem fio. Apesar de ser desenvolvido para redes sem fio, ele também pode ser empregado em redes *Ethernet* e em percursos de rede que atravessam diferentes enlaces. O diferencial do software é uma maior quantidade de métricas avaliadas, tais como taxa de perda de dados e *jitter*, bem como a sua saída de dados, que facilita a análise estatística das métricas e permite o cruzamento de dados com outras aplicações. O software ainda obtém os valores de sinal e ruído da interface sem fio, bem como o percentual de uso e de ociosidade da CPU tanto no cliente quanto no servidor. O software já foi usado em diversos artigos científicos já publicados e em outros em processo de submissão, em áreas diversas como redes sem fio definidas por software, redes veiculares e redes ad hoc.

Referências

Clark, D. (1988). The design philosophy of the DARPA internet protocols. *ACM SIGCOMM Computer Communication Review*, 18(4):106–114.

Costa, L. C., Vieira, A. B., Silva, E. B., Macedo, D. F., Gomes, G., Correia, L. H., and Vieira, L. F. (2016). Avaliação de desempenho de plano de dados OpenFlow. In *XXXIV Simpósio Brasileiro de Rede de Computadores, 2016*. SBC.

Moura, H., Bessa, G. V., Vieira, M. A., and Macedo, D. F. (2015). Ethanol: Software defined networking for 802.11 wireless networks. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 388–396. IEEE.

Teixeira, F. A., e Silva, V. F., Leoni, J. L., Macedo, D. F., and Nogueira, J. M. (2014). Vehicular networks using the IEEE 802.11p standard: an experimental analysis. *Vehicular Communications*, 1(2):91–96.

Implementação do Padrão IEEE 802.15.4e TSCH para o Network Simulator 3

Luis Pacheco[1] **, Tom Vermeulen**[2] **, Sofie Pollin**[2] **, Priscila Solis**[1]

[1]Departamento de Ciência da Computação – Universidade de Brasília
Caixa Postal 4466 CEP 70910-900 – Brasília – DF – Brasil

luisbelem@gmail.com, pris@unb.br,

KU Leuven, Dept. Elektrotechniek-esat-telemic
Kasteelpark Arenberg, 10 2444 3001 Leuven – Belgium

tom.vermeulen,sofie.pollin@esat.kuleuven.be

Abstract. *The Wireless Sensor Networks (WSNs) play a major role in defining the Internet of Things(IoT) by representing the primary platform by which computer systems will interact with the physical world to implement various solutions and applications. The IEEE 802.15.4e Timeslotted Channel Hopping (TSCH) standard is a TDMA protocol for the medium access control layer that aims to enhance the functionality of the IEEE 802.15.4 protocol to provide better support for industrial applications. This work implements the TSCH in Network Simulator 3 (ns-3). The purpose of this implementation or tool within ns-3 is to facilitate the advancement in the studies related to this protocol through an open source and collaborative simulation environment. To validate the implementation the tool was evaluated in a household scenario and the results allowed to identify the protocol's thresholds.*

Resumo. *As Redes de Sensores sem Fio (RSSI) tem um papel principal na definição da Internet das Coisas ao representar a plataforma principal pela qual os sistemas computacionais deverão interagir com o o mundo físico para implementar diversas soluções e aplicações. O padrão IEEE 802.15.4e Timeslotted Channel Hopping (TSCH) é um protocolo TDMA da camada de controle de acesso ao meio que tem por objetivo aprimorar as funcionalidades do protocolo IEEE 802.15.4 para prover melhor suporte a aplicações industriais. Este trabalho apresenta a implementação do TSCH no Network Simulator 3 (ns-3). O objetivo dessa implementação ou ferramenta dentro do ns-3 é facilitar o avanço nos estudos referentes a esse protocolo mediante um ambiente de simulação de código aberto e colaborativo. Para validar a implementação, a ferramenta foi avaliada em um cenário doméstico e os resultados permitiram identificar os valores de limiar do protocolo.*

1. Introdução

As redes de sensores sem fio (RSSI) são compostas por diversos nós que ligam o mundo físico ao mundo digital. As funções de comunicação sem fio de tais dispositivos possibilitam a transmissão dos dados adquiridos para uma unidade de processamento central, que por sua vez toma decisões de acordo com a interpretação da informação recebida.

Atualmente a variedade de áreas de aplicação de RSSIs é ampla e abrange diversos campos tais como monitoramento e controle de áreas industriais, automatização residencial, assistência médica monitorando pacientes e seus ambientes, entre outros.

Para possibilitar a comunicação sem fio em dispositivos com tais restrições o padrão IEEE 802.15.4[Group 2006] define as camadas física e de enlace. Posteriormente, em função de algumas críticas à proposta inicial, foram realizadas várias emendas. Em 2012 a emenda IEEE 802.15.4e foi lançada, adicionando, dentre outros, novos modos de operação da camada de enlace, visando aumentar a abrangência de áreas de aplicação do protocolo, principalmente para aplicações industriais.

Um dos modos de operação adicionados no 802.15.4e é o *Timeslotted Channel Hopping* (TSCH). Neste modo, os dispositivos transmitem e recebem dados de acordo com uma agenda previamente determinada pelo coordenador da rede. Também é adicionada a técnica de salto entre canais (do inglês *Channel Hopping*), que consiste na mudança das faixas de frequência (ou canais) em que as dinâmicas da comunicação pretendem aumentar a resiliência à interferência. Por exemplo, se um canal é afetado por interferência e a comunicação não é bem sucedida, na próxima tentativa outro canal será utilizado, aumentando as chances de sucesso da comunicação.

Este trabalho descreve a implementação do modelo TSCH no simulador de redes ns-3 [ns 3 project], fornecendo uma ferramenta de utilidade para avaliação e estudos de tal protocolo dentro de um simulador de código aberto e colaborativo. O TSCH possibilita a utilização de redes de sensores determinísticas na Internet das Coisas, o que se une a iniciativas similares de outros trabalhos que também buscam tal integração, como o 6TiSCH, que integra o TSCH ao IPv6 [Dujovne et al. 2014].

O presente trabalho está estruturado da seguinte maneira: a Seção 2 apresenta o protocolo TSCH em detalhes, evidenciando suas funcionalidades. A Seção 3 descreve os elementos principais utilizados para a implementação do TSCH dentro do ns-3. A Seção 4 apresenta um cenário de avaliação e discute os resultados obtidos para validar a implementação. Por fim, a Seção 5 apresenta as conclusões e lista os possíveis trabalhos futuros.

2. Timeslotted Channel Hopping (TSCH)

O protocolo IEEE 802.15.4 foi concebido em 2003[Group 2003], definindo as camadas física e de enlace para Redes Sem Fio Pessoais de Baixas Taxas de Transmissão (do inglês *Wireless Personal Area Network* (WPAN)). Posteriormente, várias revisões e emendas foram lançadas para este protocolo, visando aperfeiçoar e expandir suas funcionalidades. A emenda 802.15.4e[Group 2012] de 2012 foi especificada com a intenção de tornar o padrão adequado para outras áreas, tais como aplicações industriais e comerciais. A fim de manter a compatibilidade com os dispositivos atuais foram propostas apenas modificações na camada de enlace.

Nessa alteração são definidos três novos modos operacionais: Redes Determinísticas de Baixa Latência (do inglês *Low Latency Deterministic Network* (LLDN)), Extensão Determinística e Síncrona com Múltiplos Canais (do inglês *Deterministic and Synchronous Multi-Channel Extension* (DSME)) e Segmentação no Tempo com Salto entre Canais (do inglês *Timeslotted Channel Hopping* TSCH). A estrutura do quadro

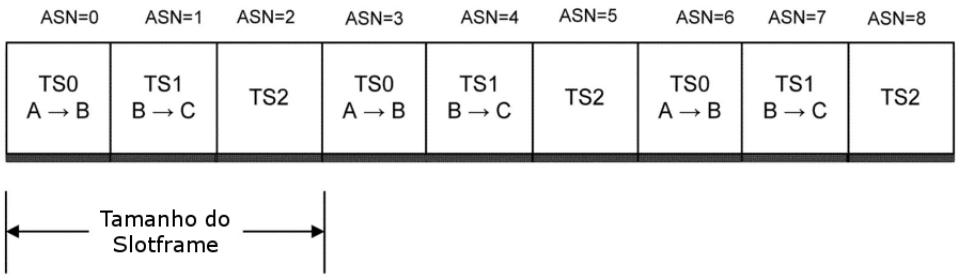


Figura 1. Estrutura do *slotframe* TSCH[Group 2011]

também foi atualizada para suportar os novos modos operacionais e proporcionar um maior nível de flexibilidade para encapsular informações.

O TSCH é um modo de operação determinístico, todas as comunicações são previamente agendadas durante a formação da rede ou configuradas anteriormente nos dispositivos. A emenda 802.15.4e não define um esquema para alterar sua configuração em tempo de execução, deixando essa responsabilidade para as camadas superiores. Um *slotframe* é a agenda de transmissões de todos os dispositivos da rede que se repete no tempo. Cada *slotframe* é composto por intervalos de tempo (*timeslots*) que determinam uma ligação entre dois (ou mais) dispositivos. Um único *timeslot* deve ser longo o suficiente para garantir uma transmissão e uma confirmação que no padrão tem uma duração de 10ms. A Figura 1 ilustra um *slotframe* configurado com três intervalos de tempo: o primeiro intervalo de tempo tem uma ligação entre os nós A e C, o segundo intervalo de tempo tem uma ligação entre os nós B e C e o terceiro intervalo de tempo está vazio.

O TSCH utiliza *Channel Hopping*, uma técnica usada para aumentar a resiliência da rede. Dispositivos saltam entre bandas de frequência predefinidas (canais), dessa forma se há alguma interferência em um canal o próximo pode estar em melhores condições, aumentando as chances de uma transmissão bem sucedida. Para certificar-se de que os dispositivos se comunicam no mesmo canal uma lista de canais é utilizada. A lista de canais pode ser configurada anteriormente, de forma aleatória ou enviada pelo coordenador. O canal a ser utilizado em uma determinada transmissão é definido de acordo com um contador global chamado Número Absoluto do Segmento (do inglês *Absolute Slot Number* (ASN)).

Também é possível que um único *timeslot* possua múltiplas ligações, neste caso os dispositivos transmissores executam um algoritmo CSMA/CA para concorrer pelo canal. A utilização do *Channel Hopping* possibilita a coexistência de configurações simultâneas do *slotframe*, onde cada nó possui um deslocamento de canal diferente. A Figura 2 apresenta uma configuração simples de múltiplos *slotframes*, neste exemplo o *slotframe* 1 possui 5 *timeslots*, e o *slotframe* 2 possui 3 *timeslots*. Múltiplos *slotframes* podem ser utilizados para definir diferentes esquemas de comunicação para diferentes grupos de dispositivos. O uso do deslocamento de canal permite que diferentes canais possam ser utilizados no mesmo ASN, isto permite que múltiplos *slotframes* ocorram simultaneamente sem colisões ou interferências. Embora cada *slotframe* possua diferentes ligações e quantidade de *timeslots*, todos eles possuem a mesma configuração de temporização.

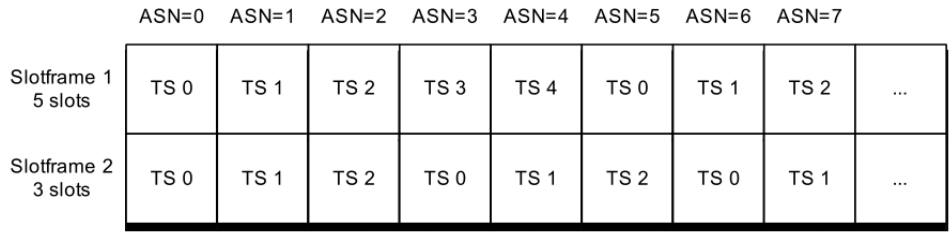


Figura 2. Múltiplos *slotframes*[Group 2011]

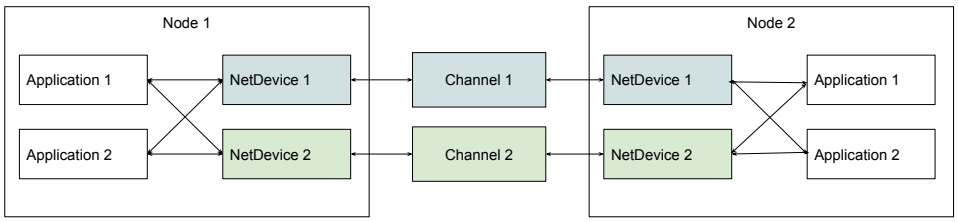


Figura 3. Arquitetura do ns-3

3. Implementação no Network Simulator 3

O ns-3 é um simulador de redes de código-aberto desenvolvido pela comunidade acadêmica que utiliza C++ para desenvolvimento de modelos e simulações e Python para o desenvolvimento de simulações.A arquitetura do ns-3 é flexível e modular, vários modelos são oferecidos e podem ser combinados a outras bibliotecas ou programas para a visualização e análise dos dados. A Figura 3 exibe a representação de dispositivos pelo ns-3 em que a classe *Node* pode possuir diversas aplicações e modelos de placas de rede.

A classe *NetDevice* faz a abstração de uma placa de rede. Cada *NetDevice* também pode ser ligado a um canal diferente. A classe *NetDevice* pode ser especializada de acordo com a tecnologia. O ns-3 já oferece algumas especializações utilizadas com frequência em simulações de rede, como *WifiNetDevice*, que implementa as camadas física e de enlace de um dispositivo Wi-Fi. Este trabalho utiliza a classe *LrWpanNetDevice*, que implementa um dispositivo que suporta o padrão IEEE 802.15.4.

A comunicação entre os dispositivos é modelada pela classe *Channel*, que emula o meio no qual a informação é transmitida. Esta classe pode ser especializada de acordo com a tecnologia a ser utilizada. O ns-3 já oferece algumas especializações, tais como o *WifiChannel*, que modela o espectro de um dispositivo Wi-Fi e o *PointToPointChannel*, que modela uma conexão ponto a ponto. O modelo *SpectrumChannel* é o utilizado neste trabalho para modelar um espectro dependente de frequência. Os canais são criados em modo global, assim todos os dispositivos utilizam o mesmo canal ao transferir dados.

Apesar de vários grupos trabalharem atualmente na implementação do IEEE 802.15.4 para ns-3 [Gholami et al. 2103], não existe ainda uma implementação que atualmente suporte o TSCH ou qualquer funcionalidade da emenda IEEE 802.15.4e[Pacheco et al. 2016] dentro deste simulador e nesse sentido, a presente ferramenta se apresenta como uma contribuição importante.

A fim de suportar os vários novos recursos introduzidos pela emenda IEEE 802.15.4e, a estrutura do cabeçalho da camada de enlace foi atualizada. Embora novos

campos tenham sido introduzidos, a compatibilidade com o padrão anterior foi mantida mediante espaços previamente reservados. Este trabalho estende a classe *LrWpanMacHeader* para suportar a nova estrutura de cabeçalho, acrescentando Elementos de Informação (EI) e a supressão de números sequenciais.

O protocolo de formação de rede não foi implementado neste trabalho, principalmente por depender de funcionalidades do 802.15.4 que não estavam presentes no modelo utilizado. O pacote de sincronização também não foi implementado, mesmo que erros de sincronização possam ocorrer em um ambiente real. Entretanto, na implementação o sistema de programação do ns-3 administra a temporização da simulação e todos os dispositivos permanecem perfeitamente sincronizados.

A emenda introduz quatro serviços em relação ao modo TSCH. O único serviço não implementado neste trabalho é o *MLME-KEEP-ALIVE*, responsável por manter os dispositivos sincronizados. Os serviços *MLME-SET-SLOTFRAME* e *MLME-SET-LINK* foram implementados completamente e são responsáveis por adicionar, modificar e remover *slotframes* e ligações, respectivamente. O serviço *MLME-TSCH-MODE* é usado para iniciar e parar o modo TSCH.

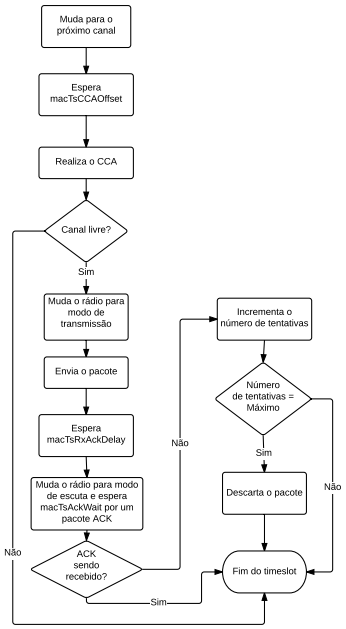


Figura 4. Fluxograma de um *timeslot* de transmissão.

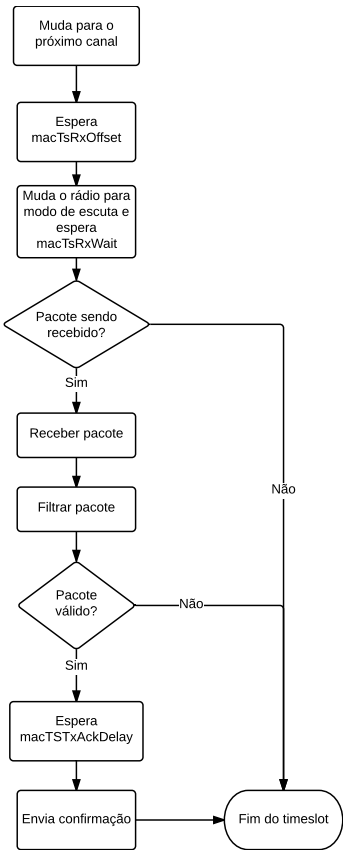


Figura 5. Fluxograma de um *timeslot* de recepção.

A Figura 4 apresenta o fluxograma de uma transmissão. O dispositivo verifica se o canal está ocioso (através do *Clear Channel Assessment* (CCA)), e caso esteja, a transmissão é realizada. Por fim o dispositivo entra em modo de escuta aguardando a confirmação. Caso o canal esteja ocupado o dispositivo tenta transmitir novamente no

próximo *timeslot*. A Figura 5 exibe o fluxograma de uma recepção. O dispositivo escuta o pacote por uma determinada duração, caso o pacote seja recebido uma confirmação é enviada, caso contrário o dispositivo volta ao estado de inatividade.

Para fornecer suporte completo à interferência, a classe *LrWpanSpectrumInterference* foi criada e integrada à classe *LrWpanPhy*, baseada na classe *SpectrumInterference* fornecida pelo ns-3. Também foi implementado um canal AWGN, em que todos os outros sinais são interpretados como interferência. A classe *LrWpanErrorModel* foi atualizada para ser compatível com a classe de interferência, o método *EvaluateChunk* é invocado sempre que as condições do canal são alteradas e o dispositivo está recebendo um pacote.

4. Cenário de Avaliação e Análise de Resultados

Para validar a ferramenta foram desenvolvidas duas simulações, disponíveis na pasta *scratch* do ns-3, são elas:

tsch-simple simula uma rede com um *slotframe* em que um dispositivos possui um *timeslot* de transmissão para enviar dados ao coordenador. A Figura 6 exibe a vazão obtida ao executar esse exemplo com diferentes tamanhos de pacotes, os resultados são comparados com valores calculados;

tsch-interference simula duas redes TSCH em execução ao mesmo tempo, causando interferência uma na outra;

A ferramenta e os manuais estão disponíveis no site *http://comnet.unb.br/br/download* e o vídeo demonstrando a instalação e utilização está disponível em *https://www.youtube.com/watch?v=D2Fa9aPGWZc*. Para a demonstração no salão de ferramentas será necessário um computador pessoal.

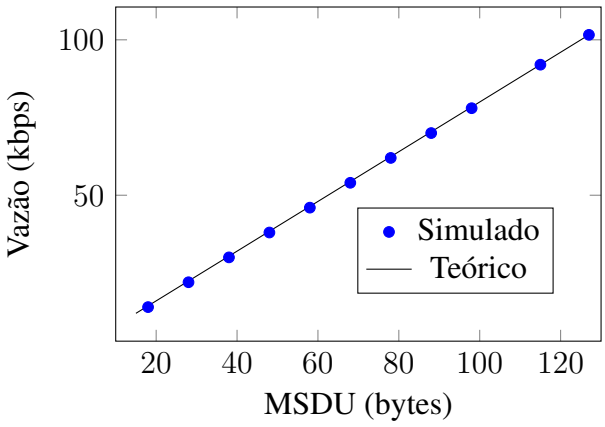


Figura 6. Vazão da rede para diferentes tamanhos de pacotes.

Para avaliar a ferramenta em um cenário comum, foi especificado um ambiente que simula uma rede residencial, que pode ter diversas aplicações tais como a economia de energia (por exemplo desligando as luzes de um quarto vazio). A Figura 7 descreve esse cenário como uma residência que possui dimensões de 8 por 11 metros e tem 5 cômodos. A rede de sensores possui 16 dispositivos, um sendo o sorvedouro, que recebe e processa todas as informações geradas pelos outros 15. A distância entre os sensores e o nó sorvedouro varia entre 1,09 metros e 6,52 metros.

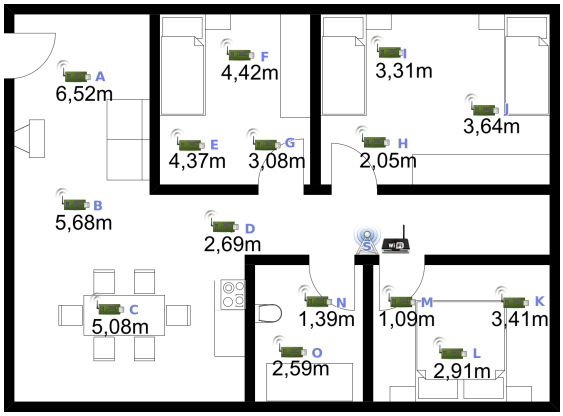


Figura 7. Residência utilizada como cenário de simulação

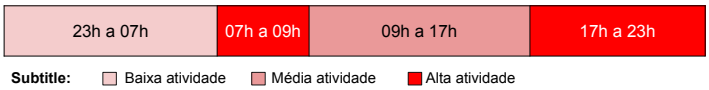


Figura 8. Atividade da rede de sensores durante o dia

A rede foi configurada com um *slotframe* de 15 *timeslots* de 10ms cada, um para cada sensor se comunicar com o nó sorvedouro. Os sensores produzem 114 *bytes* de dados (máximo permitido considerando um cabeçalho de 13 bytes) na mesma frequência de seu *timeslot*, a frequência ou o tamanho dos dados gerados não se altera a menos que o sensor fique inativo (por exemplo, no caso de um comodo vazio). Nesse caso o sensor não envia dados. Todos os parâmetros de configuração são definidos com base no padrão definido pelo protocolo. O objetivo do cenário é representar um dia comum de trabalho de uma família: o número de pessoas na residência varia, assim como a atividade da rede de sensores e do dispositivo Wi-Fi. A Figura 8 representa o cenário e os horários de maior intensidade na utilização da rede de sensores.

A Figura 9 exibe os valores de vazão em todas as regiões de intensidade do cenário simulado. Todos os dispositivos realizam a transmissão com sucesso e a vazão obtida é conforme a calculada. A Figura 10 exibe os valores de atraso os quais são comparados com o atraso máximo, que corresponde à duração de um *slotframe*, neste caso 150ms. Os resultados obtidos evidenciam uma pequena variação, relacionada ao tempo de chegada dos pacotes no *buffer* dos dispositivos.

Os resultados obtidos permitem validar a implementação do protocolo e denotam os seus limites superiores. As regiões de alta intensidade representam o uso total da capacidade do protocolo, pois o tamanho dos pacotes é o máximo e todos os *timeslots* são ocupados. A vazão encontrada foi de 101,6 kbps, valor que representa a vazão máxima a ser obtida pelo protocolo.

5. Conclusão

Este trabalho apresenta a implementação e validação do protocolo TSCH da emenda IEEE 802.15.4e no simulador ns-3 que se apresenta como uma ferramenta potencial para suporte de cenários de simulação de IoT. A natureza determinística do protocolo possibilitou a sua validação por meio de análise empírica. A disponibilização desta ferramenta

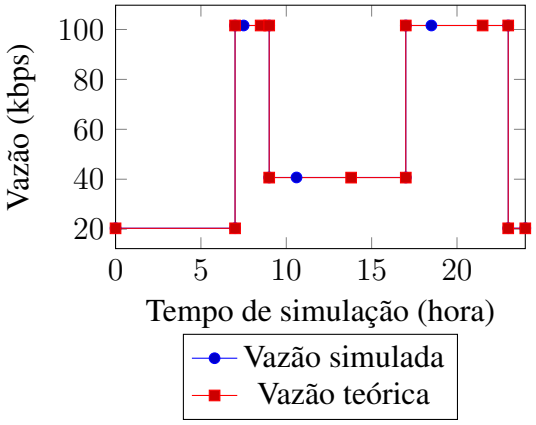


Figura 9. Vazão da rede simulada.

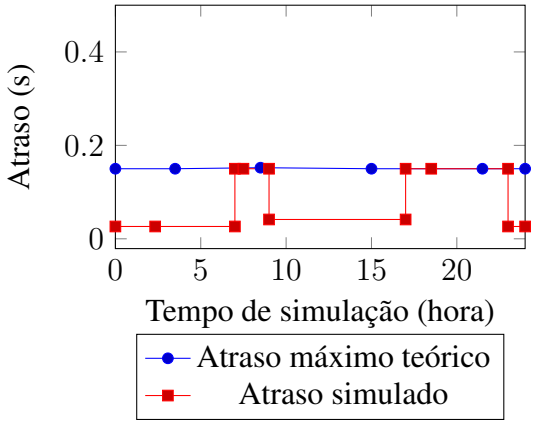


Figura 10. Atraso da rede simulada.

para uso da comunidade permitirá avaliar outros cenários e melhorar a sua implementação em simuladores de código aberto.

Referências

Dujovne, D., Watteyne, T., Vilajosana, X., and Thubert, P. (2014). 6tisch: deterministic ip-enabled industrial internet (of things). *IEEE Communications Magazine*, 52(12):36–41.

Gholami, K. E., Elkamoun, N., Hou, K. M., Chen, Y., Chanet, J.-P., and Li, J. (2103). A new wpan model for ns-3 simulator. In *New Information Communication Science and Technology for Sustainable Development: France-China International Workshop*.

Group, I. P. W. (2003). *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*. IEEE Standard for Information Technology.

Group, I. P. W. (2006). *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*. IEEE Standard for Information Technology.

Group, I. P. W. (2011). *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*. IEEE Standard for Information Technology.

Group, I. P. W. (2012). *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs) Amendment 1: MAC sublayer*. IEEE Standard for Information Technology.

ns 3 project. ns-3 network simulator. `http://www.nsnam.org/`.

Pacheco, L., Vermeulen, T., Pollin, S., and Solis, P. (2016). Evaluation of tsch/ieee 802.15.4e in a domestic network environment. In *Internet of Things. IoT Infrastructures: Second International Summit, IoT 360° 2015, Rome, Italy, October 27-29, 2015. Revised Selected Papers, Part I*, pages 257–262. Springer.

SenSE – Sensor Simulation Environment: Uma ferramenta para geração de tráfego IoT em larga escala

Ivan Zyrianoff, Fabrizio Borelli, Carlos Kamienski

Universidade Federal do ABC (UFABC)
ivan.dimitry@aluno.ufabc.edu.br,
{fabrizio.borelli,cak}@ufabc.edu.br

Resumo. *Soluções têm sido propostas para gerenciar dispositivos IoT em ambientes complexos, principalmente em Cidades Inteligentes. Embora escalabilidade seja uma característica imprescindível nesses sistemas, não existe uma plataforma de testes adequada para avaliar e comprovar a eficiência e escalabilidade desses softwares. Além disso, os cenários propostos e testados tendem a ser simples e pontuais. O SenSE é um gerador de dados sintéticos de sensores de código aberto, flexível e genérico, desenvolvido para simular ambientes complexos, como os encontrados em Cidades Inteligentes. A ferramenta é capaz de gerar uma grande quantidade de dados de sensores heterogêneos simultaneamente, sendo capaz de simular dezenas de milhares de sensores.*

Abstract. *Solutions have been proposed to manage smart devices in complex IoT environments, particularly in Smart Cities. Although scalability is an essential feature in these systems, there is not a suitable testbed to analyze and verify the performance and scalability of these platforms. Furthermore, the scenarios proposed and tested tend to be simple and limited. SenSE is an open source synthetic traffic workload generator, developed to simulate complex environments, common in Smart Cities. The tool can generate a huge amount of heterogeneous sensor data simultaneously, being able to simulate tens of thousands of sensors.*

1. Introdução

Nas últimas décadas, a humanidade passou por um rápido processo de urbanização, que trouxe diversos benefícios para indivíduos que optaram por residir em cidades grandes. No entanto, a aglomeração de pessoas ocasiona diversos problemas intrínsecos às cidades, como: trânsito, criminalidade, poluição e eliminação de dejetos. São necessárias novas soluções para esses problemas emergenciais de natureza pública. O conceito de Cidades Inteligentes prevê ambientes nos quais a tecnologia da informação seja uma ferramenta para resolver alguns desses problemas. Para isso, é necessário uma rede interconectada de dispositivos que monitoram a cidade, conhecida como de sensoriamento urbano [Zheng, et al 2014].

A Internet das Coisas (IoT) desempenha um papel fundamental na implantação de Cidades Inteligentes, viabilizando a coleta de dados de uma enorme quantidade de sensores, a análise de dados, tomada de decisões que alteram o comportamento de sistemas em tempo real. O uso inteligência computacional em conjunto com IoT, big data e computação sensível ao contexto permite a alteração dos comportamentos na velocidade necessária sem a intervenção humana direta [Perera et al. 2014]

Apesar de existirem diversas propostas de *middleware* e sistemas de processamento em tempo real para IoT e Cidades Inteligentes, não existe uma plataforma de testes

adequada para comprovar a eficiência e escalabilidade desses softwares, principalmente em cenários envolvendo milhares de dispositivos conectados. Para suprir essa necessidade, foi desenvolvido o SenSE (S*ensor Simulation Environment*), um gerador de tráfego sintético de sensores para simular ambientes complexos de IoT. Esse gerador permite simular diversos tipos de sensores, que enviam mensagens simultaneamente utilizando um protocolo de comunicação de IoT. O SenSE é flexível, o que permite que sensores diferentes sejam configurados pelo usuário, mas também disponibiliza uma série de *templates* de sensores pré-configurados para simular cenários típicos de Cidades Inteligentes.

O SenSE possui fácil implantação, podendo ser executada em praticamente qualquer computador. Uma vez instalada, através de sua interface web é possível executar e monitorar experimentos. O SenSE tem sido usado extensivamente para testar diversas implementações de sistemas de gerenciamento cientes do contexto e com *big data analytics* para IoT. Uma avaliação abrangente de desempenho foi realizada usando o SenSE, que se mostrou adequado para gerar altos níveis de carga de trabalho representativa de dados de sensores IoT.

O restante do artigo está organizado como segue: na Seção 2 estão presentes os trabalhos relacionados; a Seção 3 apresenta o SenSE, a forma como o tráfego é gerado e sua interface gráfica; na Seção 5 é descrita a demonstração da plataforma e por fim são apresentadas as conclusões do trabalho.

2.Geradores de Tráfego de Internet das Coisas para Cidades Inteligentes

Existem vários geradores de tráfego descritos na literatura, como o GlobeTraff [Katsaros et al. 2012]. O GlobeTraff permite a criação de um tráfego sintético e misto e suporta diversos tipos de tráfegos de aplicações, gerado via modelos matemáticos propostos na literatura, permitindo uma parametrização dos modelos e composição de um conjunto de tráfegos. Em comparação com o SenSE, o tráfego gerado pelo GlobeTraff não inclui dados de sensores e o SenSE permite gerar um tráfego misto com um conjunto de diferentes tipos de sensores.

Crepaldi et al. 2007 propuseram uma ferramenta de gerenciamento conhecida como SignetLab, que permite que a plataforma interaja e receba os dados dos nós da rede, preenchendo um *gap* entre as soluções de ambiente de experimentação físico (*testbed*). Rossetto et al. 2015 desenvolveu um ambiente de experimentação físico, conhecido como CeuNaTerra, para aplicações de Redes de Sensores sem Fio (RSSF) e Internet of Things (IoT) na qual permite experimentar aplicações de sensoriamento. Ambas diferenciam do SenSE já que os sensores não são abstraídos e os nós fisicamente existem, e portanto não é possível instanciar uma grande número de dispositivos. Além disso, ela não se aplica ao ambiente de Cidades Inteligentes e não permite a criação de novos sensores, como o SenSE.

3. SenSE – Sensor Simulation Environment

A ferramenta SenSE é um gerador de tráfego sintético de sensores de código aberto, com o objetivo de simular o tráfego de dados em ambientes IoT, especialmente em Cidades Inteligentes. O SenSE é uma plataforma de testes adequada para comprovar a eficiência e escalabilidade de sistemas de *middleware* em tempo real e possui a capacidade de gerar tráfego correspondente a carga de trabalho de milhares de sensores enviando mensagem, simultaneamente. O código-fonte juntamente com um manual de instalação e um tutorial

em vídeo[1] estão disponíveis no github[2]. Após o *download* e execução da ferramenta há uma interface web que além das funcionalidades da plataforma, oferece tutoriais e sua apresentação. Há também um *branch* no github da ferramenta que permite sua execução por linha de comando, sem utilizar a interface gráfica.

A ferramenta foi desenvolvida em Java, possui instalação simples e não requer muitos pré-requisitos para seu funcionamento. A execução do SenSE é intuitiva, sendo necessário apenas designar um endereço IP válido, definir o tempo de experimento e configurar a quantidade e os tipos de sensores desejados. O envio de mensagens pela ferramenta é realizado pelo MQTT[3], um protocolo M2M (*machine-to-machine*) de camada de aplicação típico de ambientes IoT, desenvolvido para sensores e dispositivos móveis. O MQTT utiliza o paradigma *publish/subscribe* para a troca de mensagens, no qual um cliente publica uma mensagem em um tópico e todos os outros clientes que são subscritos nesse tópico irão receber a mensagem. Para isso, é necessário um *broker*, responsável por receber as mensagens e enviá-las corretamente para os clientes inscritos. Foi utilizada a biblioteca Eclipse Paho[4] para enviar mensagens através do protocolo MQTT.

O estudo de sensores em ambientes IoT identificou dois grandes grupos de sensores que se diferenciam pelo modo de gerar os dados:

- Sensores movidos por tempo: enviam dados periodicamente para reportar um estado (ex.: um sensor de qualidade do ar que envia dados a cada 30 minutos);
- Sensores movidos por evento: enviam dados caso o seu estado atual seja alterado (ex.: um sensor de presença que envia dados quando alguma presença é detectada).

Ambas as categorias contribuem para gerar um ambiente heterogêneo, característico de redes IoT, sendo que o tráfego de sensores movidos por tempo possui maior regularidade e o tráfego gerado por sensores movidos a evento tem natureza sazonal, com momentos de pico e períodos de envios escassos.

A modelagem de sensores movidos por evento possui maior complexidade, uma vez que é necessário uma mudança no seu estado gerada por algum fator externo. Considerando um cenário de Cidades Inteligentes, foram modelados sensores cujo acionamento é causado pelo comportamento humano e a modelagem é feita para simular pessoas chegando a um evento (ex.: chegadas em um estádio) e a chegada de um indivíduo modifica o estado do sensor, gerando uma mensagem. Ao contrário de sensores movidos por tempo, não se define uma quantidade de sensores movidos por evento, mas uma taxa de chegada de pessoas (indivíduos por segundo).

A estrutura da mensagem enviada pelos sensores foi baseada em uma implementação em um cenário real, utilizada no sistema IMPReSS [Kamienski et. al 2016], um gerenciador de contexto para ambientes IoT. A mensagem possui o seguinte padrão:"`type=sensorType;resource=sensorID;message=sensorMessage;P1=time stamp;"`. O tipo do sensor refere-se a informação que ele envia, o *resource* é o ID do sensor, *message* é o campo onde estão as informações enviadas por aquele sensor e P1 é o *timestamp* de quando essa mensagem foi enviado.

Para a realização de experimentos é necessário que o usuário especifique os sensores que deseja gerar tráfego. Os atributos configuráveis são:

[1] https://www.youtube.com/watch?v=8Kq3Z3NkW4A&t=6s
[2] https://github.com/ivanzy/SenSE-Sensor-Simulation-Environment
[3] http://mqtt.org/
[4] https://eclipse.org/paho/clients/java/

- Tipo do sensor: nome do tipo do sensor, presente no campo *type* da mensagem, normalmente o nome do tipo de sensor tem relação com a informação (ex.: um sensor que envia dados de temperatura teria seu tipo como *temperature*);
- Tópico: tópico MQTT que aquele tipo de sensor vai publicar as mensagens;
- Tipo do dado: o tipo de dado que será o enviado pelo sensor, pode ser escolhido entre tipos básicos (int, *float*, *boolean* e *char*) e, caso o usuário desejar, um valor máximo e mínimo para aquele tipo de sensor , as mensagens daquele tipo de sensor serão aleatórias dentre o tipo escolhido (ex: definido 100 e 50 como máximo e mínimo, todas as mensafens gesão um valor inteiro aleatório nesse intervalo). Além dos tipos básicos também é possível escolher *booleanText*, esse caso o *payload* será sorteado entre a *String* que for digitada no campo max e no campo min (ex: se o max for "on" e o mínimo "off" o *payload* será sorteado, a cada mensagem enviada, entre "on" ou "off").

Além disso, há especificações adicionais diferentes para sensores movidos por tempo e sensores movidos por evento. Para sensores movidos por tempo:

- Periodicidade: a periodicidade com que sensores enviam dados (ex.: a cada 30 segundos ou a cada 6 horas);
- Número de dispositivos: número de sensores instanciados do tipo definido.

Para sensores movidos por evento:

- Modo da taxa de chegada: a taxa de chegada pode ser contínua ou variável, esse atributo será explicado com mais detalhes na seção 3.1.
- Valor da taxa de chegada: o valor, em indivíduos por segundo, para envio de mensagens.

O SenSE possuiu *templates* de sensores prontos para o uso. Estes *templates* foram desenvolvidos baseados na modelagem de Cidades Inteligentes criada na cidade Pádua na Itália [Zanella et. al., 2014] para fornecer um ambiente próximo ao encontrado em cenários reais. Serviços descritos em Zanella et. al foram transformados em tipos de sensores. Na Tabela 1 é descrito as especificações da cada *template* presente na ferramenta.

Tabela 1. *Templates* de sensores disponíveis no SenSE

Tipo de Sensor	Taxa de envio (pacotes)	Categoria
Qualidade do ar	1 mensagem a cada 30 min por sensor	Tempo
Barulho	1 mensagem a cada 30 min por sensor	Tempo
Estrutura de edifícios	1 mensagem a cada 10 min por sensor	Tempo
Congestionamento (trânsito)	1 mensagem a cada 10 min por sensor	Tempo
Gestão de resíduos	1 mensagem a cada 60 min por sensor	Tempo
Fluxo de pessoas	1 mensagem quando alguém é detectado	Evento

3.1. Geração de Tráfego Sintético

A inspiração para o método de geração de tráfego de sensores movidos por tempo de forma eficiente baseia-se no funcionamento de simuladores computacionais. Simuladores, resumidamente, possuem uma fila de tarefas, nas quais cada tarefa será realizada em um determinado tempo de simulação. Essa tarefa pode ou não criar outras tarefas na fila para serem realizadas.

O SenSE realiza algo parecido para cada tipo de sensor movido por tempo. São instanciados objetos sensores de acordo com as configurações e o número de dispositivos

especificados pelo usuário. Cada sensor possui o tempo em que irá enviar sua primeira mensagem. Esses sensores são postos em uma fila e ordenados de acordo com o tempo de envio da sua primeira mensagem, sendo que o primeiro sensor da fila será o que possuir o menor tempo de envio. O programa irá aguardar até que o momento que deve enviar a primeira mensagem do primeiro sensor da fila. Após isso, é a periodicidade definida pelo usuário adicionado ao tempo do sensor, que é inserido novamente no final da fila. O mesmo processamento é executado para todos os sensores na fila até o final do experimento. Para que seja possível executar simultaneamente vários sensores de tipos diferentes, cada fila é implementada como uma *thread* e é necessário uma fila para cada tipo de sensor diferente, definido pelo usuário. A Figura 1 ilustra o funcionamento do SenSE para sensores movidos por tempo.

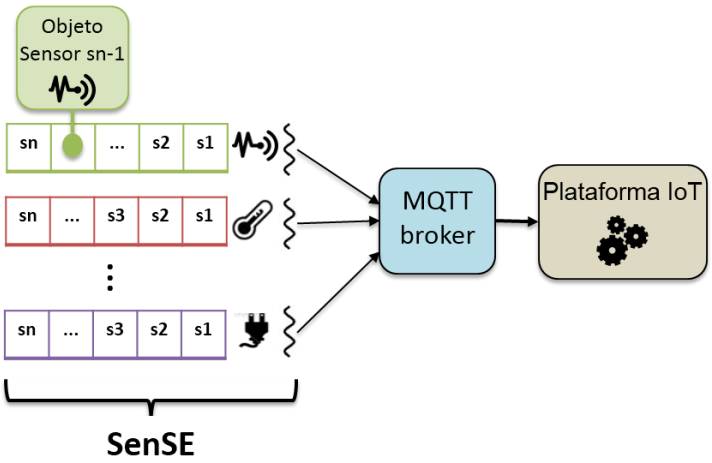


Figura 1. Funcionamento do SenSE para sensores movidos por tempo.

Sensores movidos por evento são modelados de forma diferente. A chegada de indivíduos em um evento pode ser modelada utilizando a distribuição de Poisson, e o parâmetro λ (lambda) define a taxa de chegada de pessoas a um evento simulado. Foi observado que a taxa de chegada de pessoas em uma evento típico (ex.: chegada de pessoas à um estádio) varia em três momentos: a) Início: antes e instantes após o início, com taxa de chegada moderada, uma vez que as pessoas chegam com antecedência ou um pouco atrasados ao evento em tempos diferentes; b) Meio: durante o evento a taxa de chegada é pequena, pois poucos indivíduos entram e saem do evento; c) Fim: após o término do evento a taxa de chegadas é alta pois todos os indivíduos costumam sair ao mesmo tempo. Por exemplo, quando a taxa de chegada é X indivíduos por segundo no Início, no Meio é diminuída para X/10 e no Fim ela assume 3X. Esse comportamento pode ser observado no gráfico da Figura 2, gerado a partir dos dados de log do SenSE. O evento simulado tem duração de 2/3 do tempo do experimento e inicia-se aleatoriamente do tempo 0 até o tempo máximo para que a duração do evento não extrapole o tempo do experimento.

O comportamento registrado no gráfico da Figura 2 ocorre no sensor disponibilizado como *template* e também nos sensores movidos por evento criados cujo o modo da taxa de chegada é definido como variável. Sensores movidos por evento configurados com taxa de chegada contínua enviam mensagens durante todo o experimento com o mesmo parâmetro λ. Outras distribuições podem ser adicionadas ao modelo de geração de sensores movidos por evento.

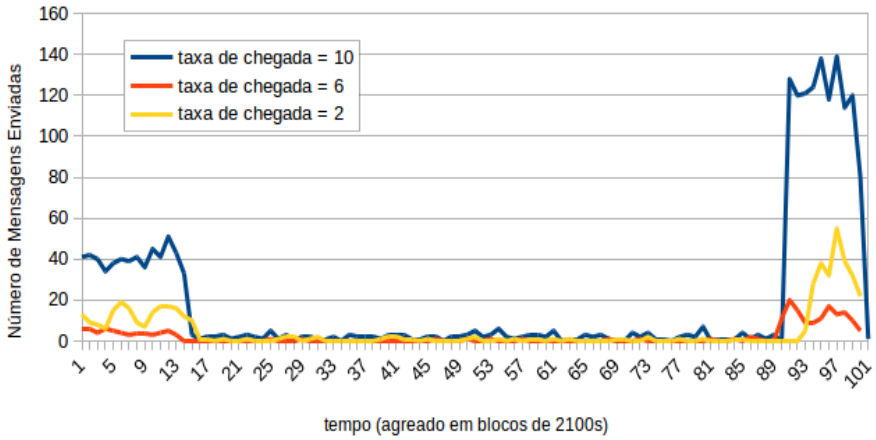


Figura 2. Comportamento de sensores movidos por evento com taxa de chegada variável

3.2. SenSE Web GUI

O SenSE possui uma interface web para interagir com o usuário. Após a instalação e execução da ferramenta, é possível acessar a GUI (*Graphic User Interface*) pelo endereço http://localhost:8080/SenSE de um *browser*. Foram aplicados conceitos de usabilidade para tornar a interface intuitiva. A *interface web* é composta por cinco páginas:

- *Home*: página inicial, contém uma apresentação ao SenSE, informações básicas e um tabela descrevendo o conteúdo de cada página web;
- Tutorial: essa página é composta de quatro tutoriais, de complexidade crescente, sobre o uso do SenSE;
- *Settings*: nessa página que são configurados experimentos, desde informações de rede até criação de novos sensores. Há botões de ajuda explicando para usuários iniciantes a utilidade de cada campo. É nessa página que o usuário inicia a execução do SenSE, clicando no botão RUN. Uma vez que o experimento inicie com sucesso, o usuário é redirecionado para página de monitoramento;
- *Monitoring*: o monitoramento do experimento é feito nessa página, que é composta por uma tabela atualizada dinamicamente com as últimas mensagens enviadas, sendo que cada sensor possuí uma cor de fundo diferente para ser melhor identificado, informações sobre o tempo do experimento (também atualizadas dinamicamente), um botão de parada do experimento executado e um gráfico que marca um ponto a cada dez segundos, com a quantidade de mensagens enviadas naquele período de tempo. Parte desta página pode ser vista na Figura 4;
- *Technical Details*: nessa seção são apresentados detalhes técnicos da plataforma.

4. Demonstração

Serão realizadas duas demonstrações da plataforma. Na primeira será explorado as diversas configurações do SenSE, na qual serão realizados experimentos rápidos com a geração de sensores movidos por tempo e por evento. Na segunda, o SenSE será utilizado para testar um plataforma IoT real, com o gerenciador de contexto IMPReSS.

4.1. Demonstração 1: recursos do SenSE

Nessa demonstração serão realizados uma série de experimentos curtos utilizando o SenSE. O tráfego dos sensores será gerado de diferentes maneiras, mas não será processado por uma outra aplicação.

Inicialmente, será apresentada uma aplicação simples (do tipo "*hello world*"), que executa um experimento com somente um tipo de sensor, escolhido dentre os templates de sensores movidos por tempo. Todas as configurações estão presentes na página "*Settings*". Primeiro são definidas as configurações básicas do experimento (endereço IP do *broker*, porta, nome do experimento e tempo do experimento). Após isso, é escolhido um tipo de *template* de sensor e define-se um tópico para o envio de mensagens desse sensor além de uma quantidade de sensores desse tipo. A Figura 3 ilustra esse processo. Em seguido, pressiona-se o botão RUN e o experimento inicia, redirecionando para a página "*Monitoring*", na qual é possível acompanhar o experimento pela tabela e gráfico, a Figura 4 mostra uma parte da tela de *monitoring*.

Figura 3. Demonstração do SenSE: aplicação de *hello world*

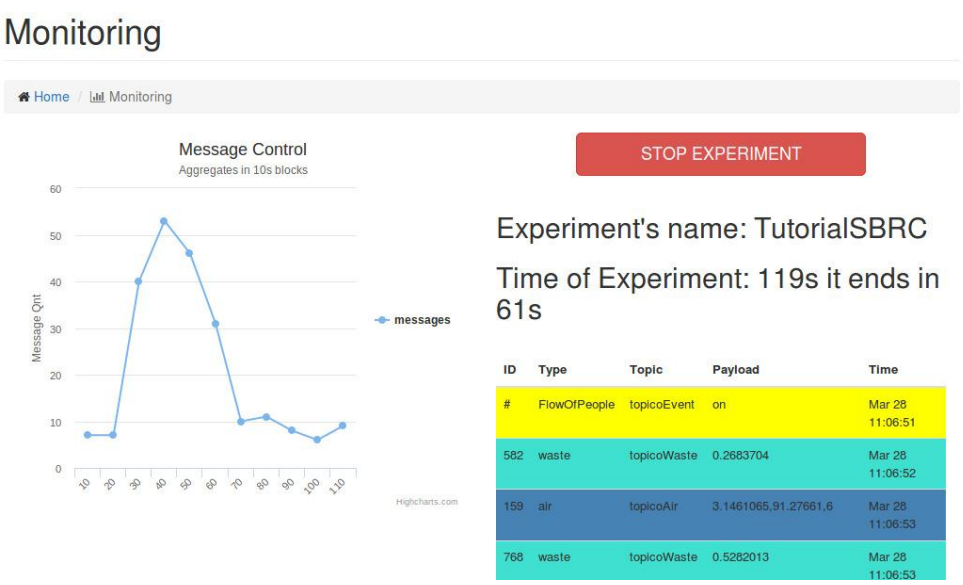


Figura 4. Tela de monitoramento do SenSE

Após essa aplicação, será realizado o mesmo processo para sensores movidos por evento, no qual será possível observar no gráfico o comportamento presente na Figura 2. Por fim, será demonstrado como criar novos sensores, além de um experimento com vários tipos de sensores diferentes enviando mensagens - esse experimento será acompanhado na tela de "*Monitoring*". O tempo planejado de cada experimento é por volta de três minutos.

4.2. Demonstração 2: utilizando o SenSE para testar uma aplicação IoT

O SenSE será utilizado para gerar carga de trabalho para uma aplicação, o gerenciador de contexto IMPReSS [Kamienski et. al 2016], desenvolvida para ambientes IoT. Previamente serão definidas um conjunto de regras no gerenciador para que as mensagens enviadas pelo SenSE acionem-nas, fazendo com que a plataforma tome uma ação, ou seja,

envie mensagens aos atuadores. A plataforma IMPReSS será executada em um computador.

Com o IMPReSS em execução, será configurado utilizando a GUI do SenSE um experimento com duração de cinco minutos, no qual será enviado para plataforma mensagens provenientes de todos os *templates*, um em cada tópico. Cada *template* de sensor movido a tempo será configurado para ter 2.500 sensores e o *template* de sensor movido por evento será configurado com taxa de chegada igual a 5. Quando as mensagens começarem a ser enviadas, iremos acompanhar o experimento pela tela de monitoração do SenSE e, usando o terminal, iremos subscrever no tópico no qual o IMPReSS envia instruções aos atuadores (fictícios) para o acompanhamento das ações tomadas pela plataforma.

5. Conclusão

A falta de uma ferramenta de geração sintético de sensores de ambientes IoT complexos levou ao desenvolvimento do SenSE. A ferramenta é de simples implantação e tem a capacidade de gerar tráfego referente a milhares de sensores diferentes simultaneamente. SenSE é uma ferramenta flexível, que permite a configuração de sensores específicos para experimentos pontuais. Como trabalhos futuros planeja-se implementar novas funções de distribuição para o envio de mensagens dos sensores movidos á evento assim como uma forma de enviar dados de sensores para a ferramenta e ela inferir a distribuição de probabilidade de envio daquele sensor, podendo gerar mensagens segundo esse comportamento.

Referências

Kamienski, C., Jentsch M., Eisenhauer M., Kiljander J., Ferrera E., Rosengren P., Thestrup P., Souto E., Andrade W., Sadok D. (2017), “Application development for the Internet of Things: A context-aware mixed criticality systems development platform”, Computer Communication, vol. 104, pp. 1-16.

Zanella, A., Bui, N., Castellani, A., Vangelista, L., Zorzi, M. (2014), "Internet of Things for smart cities", IEEE Internet Things Journal, vol. 1, no. 1, pp. 22-32.

Crepaldi, R., Friso, S., Harris, A., Mastrogiovanni, M., Petrioli, C., Rossi, M., Zanella, A., and Zorzi, M. (2007). The design, deployment, and analysis of signetlab: A sensor network testbed and interactive management tool, TridentCom 2007.

K. V. Katsaros, G. Xylomenos and G. C. Polyzos, "GlobeTraff: A Traffic Workload Generator for the Performance Evaluation of Future Internet Architectures," 2012 5th International Conference on New Technologies, Mobility and Security (NTMS), Istanbul, 2012, pp. 1-5.

Rossetto, Silvana; Silvestre, Bruno; Rodriguez, Noemi; Branco, Adriano; Kaplan, Leonardo; opes, Ian; Boing, Miguel; Santana, Douglas; Lima, Vinicius; Gabrich, Raul; "CeuNaTerra: testbed para espaços inteligentes”, 2015. SBRC 2015. Salão de Ferramentas.

Perera, C., et. al. (2014), "Context Aware Computing for The Internet of Things: A Survey", IEEE Comm. Surveys & Tutorials, 16(1), First Quarter 2014.

Zheng, Yu, Capra, L., Wolfson, O., Yang, H. (2014), "Urban Computing: Concepts, Methodologies, and Applications", ACM Transactions on Intelligent Systems and Technology.

Salão de Ferramentas do SBRC 2017

Sessão Técnica 2
Redes Definidas por Software e Orientadas a Conteúdo

IT-SDN: Improved architecture for SDWSN

Renan C. A. Alves[1], **Doriedson A. G. Oliveira**[1], **Gustavo A. Núñez**[1], **Cintia B. Margi**[1]

[1]Universidade de São Paulo – São Paulo, Brazil

Abstract. *Wireless Sensor Networks (WSN) and the Internet of Things (IoT) are promising technologies for new applications. One of the main challenges in such environments is how to enable flexibility on infrastructure. Software Defined Wireless Sensor Networks (SDWSN) has been presented as an effective approach to address several issues in WSN and IoT. There are several approaches in the literature, but unfortunately there are drawbacks for these approaches. We present IT-SDN, an open SDWSN tool, that has a clear separation of the protocols used to achieve southbound communication, neighbor discovery and controller discovery, unlike other approaches on the literature. We also demonstrate how to evaluate the main performance metrics of an IT-SDN deployment.*

1. Introduction

Wireless Sensor Networks (WSN) and the Internet of Things (IoT) are composed of devices capable of sensing/actuation, communication and processing, thus enabling the development of applications in several areas, such as health, environmental, industrial, urban monitoring. One of the main challenges in WSN is how to enable flexibility on infrastructure, considering that all nodes behave both as forwarding devices and end nodes, there are different routing patterns, different applications being executed by the nodes, and yet it should be energy efficient and limit the control traffic (both by reducing the number of control messages and by limiting the frame size to the typical IEEE 802.15.4 frame).

Software-defined networking (SDN) is an approach to manage networks that separates the control plane from the data plane. The routing/forwarding decisions are made by the SDN controller based on network information received. Openflow [McKeown et al. 2008] was the first protocol proposed to establish the communication between forwarding nodes and the SDN controller.

Software Defined Wireless Sensor Networks (SDWSN) has been presented as an effective approach to address several of these issues in WSN. There are several approaches in the literature [Kobo et al. 2017]. Unfortunately there are drawbacks for these approaches. The ones based on OpenFlow have issues concerning frame sizes, introduced overhead and use of TCP as underlying communication protocol. SDN-Wise implementation is not completely available for download and use. TinySDN provides the code and related documentation, but it is highly dependable on TinyOS, thus limiting the platforms it could be deployed. Furthermore the use of the ActiveMessage function limits interoperability with other systems.

In this paper, we describe IT-SDN, an SDWSN tool that is completely open and available, unlike SDN-Wise. While its design is inspired by TinySDN [de Oliveira et al. 2014], we improved the architecture, protocols and implementation. The underlying architecture is independent of the operating system and its functions. IT-SDN has a clear separation of the protocols used to achieve southbound

communication (SB), neighbor discovery (ND) and controller discovery (CD), unlike SDN-Wise and TinySDN, which allows evaluation of different approaches to achieve neighbor and controller discovery. Furthermore, our implementation of IT-SDN supports configuring multiple nodes with a single packet, being up to the controller whether to use it or not. This feature was envisioned in previous work but has not been implemented [de Oliveira et al. 2014, Galluccio et al. 2015].

This paper is organized as follows. Section 2 presents IT-SDN architecture and specification, while Section 3 discusses the software architecture and implementation details. The proposed demonstration is described in Section 4, while Section 5 presents final remarks and future work.

2. Architecture

In this section, we discuss our view of the SDWSN architecture and our implementation approach, considering the target operating system. Our SDWSN architecture contains three components as shown in Figure 1, namely a southbound (SB) protocol, a neighbor discovery (ND) protocol and a controller discovery (CD) protocol.

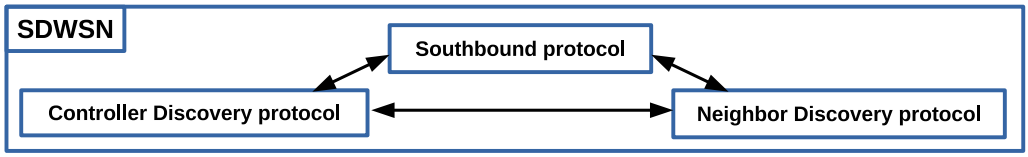


Figure 1. SDWSN architecture

The purpose of the SB protocol is to define the communication procedure between controller and SDN-enabled devices, including packet format definition, how to process these packets (Section 2.1) and the operation workflow (Section 2.2).

We devised a custom SB protocol based on TinySDN [de Oliveira et al. 2014] in the sense that we also employ the "flow id" concept, i.e. packets are routed according to a predefined label. Some control packets are routed by address, as detailed in Section 2.1. Like TinySDN, we also use an underlying protocol to perform ND and CD, but we do not demand a predefined protocol nor use it to transmit control packets.

A ND protocol obtains and maintains node neighborhood information, while a CD protocol identifies a next hop candidate to reach the controller. In the case of static networks, the former may be executed just at network bootstrapping, conversely, it should run continuously in networks with mobility and changing links. The latter is used mostly at network bootstrapping, before the node gets routing information from the controller.

We designed both ND and CD protocols as separate entities due to the existence of many suitable alternatives to theses tasks [Chen and Bian 2016]. With this approach a researcher can easily change and compare the performance of different algorithms, or design and implement a new algorithm and integrate it with IT-SDN framework; also, a developer can choose the best alternative according to the application scenario.

To provide this flexibility, a ND protocol must conform to a predefined interface, which contains two functions (initialization procedure and packet reception) and generates events upon neighborhood changes. Similarly, there is an interface for CD protocols

that provides a function to get the current best neighbor candidate to reach the controller, initialization, packet reception and event generation (in case of candidate change).

2.1. Packet types

This section lists all packet types supported by IT-SDN, their content and how each of them should be processed.

Packet header: Every packet contains a type identifier, a "time-to-live" field, a sequence number and the source address. These values compose the packet header.

Flow request: These packets are always destined to the controller, therefore are routed according to the nodes flow id table. They contain a query about how to route a certain packet, in case of a flow table miss-match. This packet has two versions: one for solving unknown flow id-oriented flows and another for solving address-based routes.

Flow setup: The purpose of this packet is to insert or change a flow table entry in a certain network node. It can be sent in response to a flow request or in accordance to the controller routing policies (for instance, if a better route is discovered). This packet contains a destination address (hence routed by the address flow table), a route identifier, the action to execute upon receiving a packet with such identifier and an action parameter. The route identifier is a flow id or an address, while the action is “drop packet”, “receive packet” or “forward packet” and the action parameter is the next hop (if applicable). The framework could be further extended with other actions.

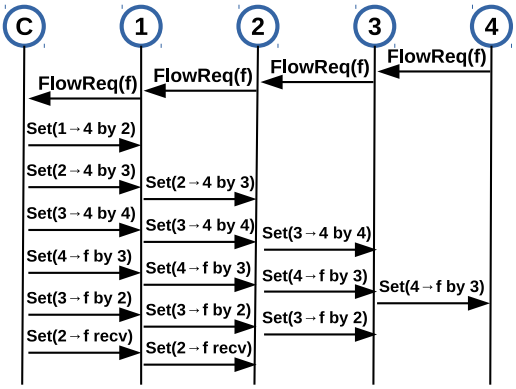
Source routed flow setup: These packets serve the exact same purpose of regular flow setups. However, flow setups are routed according to address flow tables, which have limited size. Therefore, nodes near the controller, which have to know how to reach a large number of nodes, could overflow the maximum number of entries in the address flow table in large networks. In order to avoid this undesired behavior, we added the possibility of source routing flow setup packets, bypassing the use of address flow tables. Since the controller already has a global view of the network, it is able to calculate the source route header information.

Multiple flow setup: This is a generalization of source routed flow concept, in the sense that it is able to set a route in many nodes with one single packet that is source routed (and therefore does not use address flow table). The packet is sent to the first node that should have its flow table altered, which in turn forwards the packet to the "next hop" field of the freshly configured flow table entry.

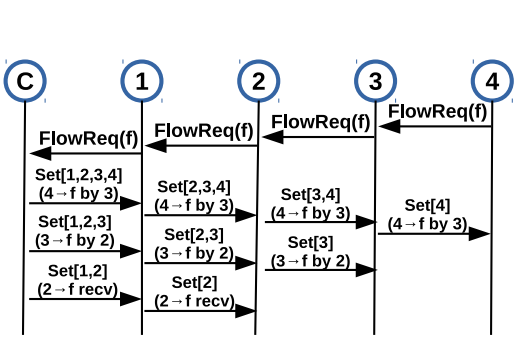
Figure 2 shows a comparison of the three flow setup possibilities, considering a 5-node linear topology (a controller and four nodes). In this example, node 4 asks how to route flow “f”, destined to node 2. Using regular flow setups, 15 control messages are transmitted or forwarded. Source routed and multiple flow setups use 9 and 6 messages, respectively.

Acknowledgement: Control packets may require delivery confirmation, what is provided by an acknowledgement packet, which relies on the sequence number.. For instance, it is used to ensure that the policy set by the controller has reached the nodes.

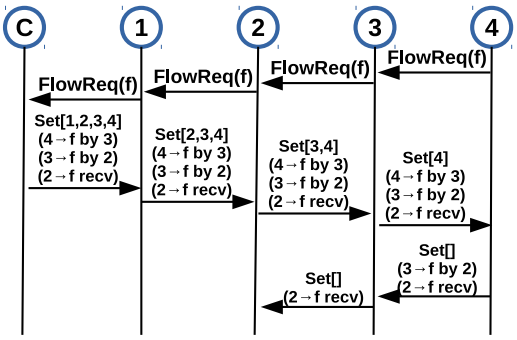
Neighbor discover / controller discovery: Packets used by the underlying ND/CD protocols. These packets are not processed by the SDN core, they are sent directly to the



(a) Regular flow setup: 15 transmissions



(b) Source routed flow setup: 9 transmissions



(c) Multiple flow setup: 7 transmissions

Figure 2. Comparison of flow setup types: setting flow "f" from node 4 to node 2

corresponding module.

Neighbor report: Nodes send this packet to update the controller network representation. It contains a list of the current node neighbors associated with a link metric (e.g. ETX, RSSI or LQI). The underlying ND protocol determines how often it is transmitted.

Data packet: Packets sent by the application layer, routed according to the flow id table.

2.2. Workflow

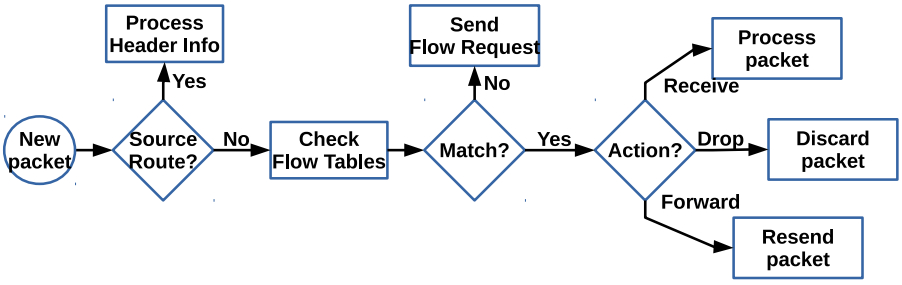


Figure 3. IT-SDN Workflow

An SDN-enabled node operates according to the workflow presented in Figure 3. Upon receiving a packet, the node checks whether it is routed by the flow tables or source routed. Source routed packets are always forwarded according to the header, while all other packets should match a routing rule in one of the flow tables. If the node does not

find a matching rule to process the packet, a flow request is sent to the controller. The node may store the packet until it gets a corresponding flow setup. If a matching rule is found, the corresponding action is executed, namely process the packet, forward it to a next hop or drop it. Other actions could be included.

On IT-SDN, the main sources of energy consumption are route requesting and route maintenance. Route requesting is based on the workflow above, and its overhead is decreased by the enhanced flow setup packets presented in Section 2.1. The larger the distance between node and controller, more energy is spent to configure a route. This can be alleviated by using multiple controllers, which could be closer to the nodes. Route maintenance is tightly connected to the underlying ND protocol, as it is responsible for detecting link status. Fine tunning ND message interval and link change threshold may greatly impact energy consumption.

3. Software Architecture

Considering the general architecture presented in Section 2, we instantiated a compliant software architecture targeting Contiki OS [Dunkels et al. 2004], an open source lightweight operating system for the Internet of Things that supports many devices (such as TelosB and SensorTag).

The software architecture components exhibited in Figure 4 maps to the three main components of the general SDWSN architecture: ND and CD protocols are clearly distinguished, while `process packet` and `core` cover the SB protocol.

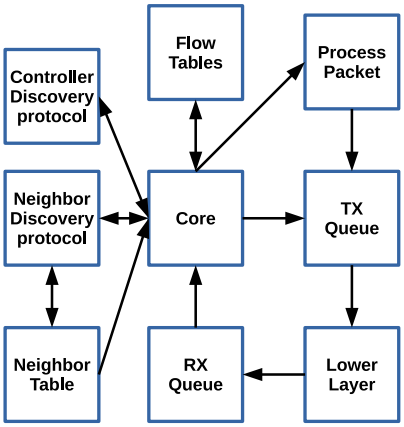


Figure 4. IT-SDN software architecture

Additionally there are four auxiliary components, namely the flow tables, a neighbor table and RX/TX queues. The flow tables store routing information set by the controller, divided into two tables: flow id table and address flow table. Their fields were already discussed in Section 2.1, with the addition of two columns, the number of times the entry has been used and the age of entry. These extra fields are used to decide which entry to overwrite in case the flow table overflows. An example is provided in Figure 5.

The neighbor table provides a common data structure for the ND protocol to write discovered information, so the SDN core can read it and transmit neighbor report packets. The RX/TX queues are buffers to avoid dropping packets in case of network congestion or long delays on packets processing.

Flow id OR Address	Action	Action Parameter	# uses	age
Flow id: 0	forward	4	4	3
Flow id: 5	receive	-	10	2
Address: 0x1234	drop	-	2	1

Figure 5. Example of flow table

The `core` component orchestrates all other modules on an event based fashion as shown in Figure 6. Neighbor report packets are sent to the controller upon an event from ND protocol. The CD protocol event is used to set the controller route for the first time. It uses a reserved flow id number to identify the controller flow. With this convention, nodes can send requests to the controller without knowing its actual address or the actual number of controllers.

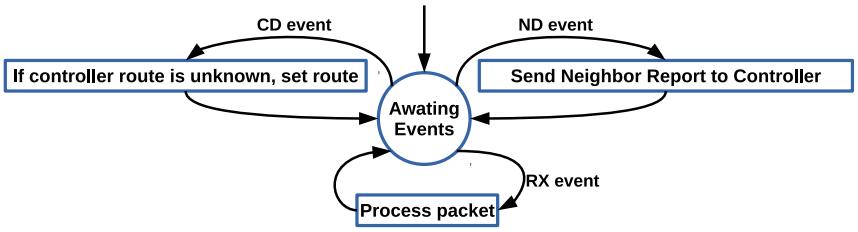


Figure 6. IT-SDN core operation

Received packets are processed as explained in Sections 2.1 and 2.2. It first checks the packets routing category (flow id, address or source routed) and then performs the associated action. This component also keeps a list of packets that are awaiting an appropriate flow table entry.

3.1. Implementation Details

We implemented the SDWSN protocol as a Contiki *network driver*, using a C language *struct* that abstracts layer 3 protocols. Our implementation could replace the existing layer 3 protocols available in the OS (IPv6 and RIME stack). In order to use Contiki's implementation of the Collect Protocol[1] as the SDN ND/CD protocols, we made our implementation compatible to the RIME stack by adding a compatibility byte to the SDN packet header (`github.com/contiki-os/contiki/blob/master/core/net/rime/collect.c`). Additionally, we use Contiki Link Layer Security (LLSEC) stack as a decision point to forward packets to SDN or RIME stack. We chose this approach to avoid modifying Contiki's code.

3.2. Controller

We provide a controller software along with the SDN-enabled node code. A Contiki node is used as a bridge between the wireless network and the controller software running at a PC. The Contiki node runs ND and CD related tasks, in addition to performing radio transmission and receptions

[1] `github.com/contiki-os/contiki/blob/master/core/net/rime/collect.c`

in behalf of the PC. The communication between PC and Contiki node is achieved through a serial (USB) connection (or TCP connection in case of COOJA simulations).

The PC software maintains a centralized global view of the network stored as a graph, according to the neighbor report packets received from the network nodes. This representation is used to calculate flow requests from the nodes, and to update flow tables according to Dijkstra shortest path algorithm. The flow table cache maintains a copy of routes previously configured on the nodes. Neighbor report messages may change the controller network view and trigger route recalculations. The flow table cache is used to avoid sending flow setup messages of unchanged routing rules. For instance, if a previous route $A \rightarrow B \rightarrow C \rightarrow D$ is recalculated to $A \rightarrow N \rightarrow C \rightarrow D$, node C flow table does not need to be updated.

3.3. IT-SDN website

IT-SDN is available at `http://www.larc.usp.br/~cbmargi/it-sdn`, including the source code and its license. The website also includes the installation and developer manuals. In order to run IT-SDN, it is necessary to use Contiki 3.0 and mspgcc LTS 20120406 (4.6.3), as detailed in the installation manual. The controller runs on Linux (we tested on Ubuntu 16.04 64bit) with Qt 5.8.

4. Demo description

This demonstration has tree main steps. First we will show how to program a WSN application using IT-SDN and run it on the COOJA [Osterlind et al. 2006] environment, emulating TelosB[2] mote. Next we will present an overview of IT-SDN main features, including CD, ND and SB message exchanges, which will create the necessary flows to deliver data from the previously created application. Lastly we will show how to evaluate the performance of an IT-SDN deployment, obtaining end-to-end delay, delivery rate, and control plane overhead.

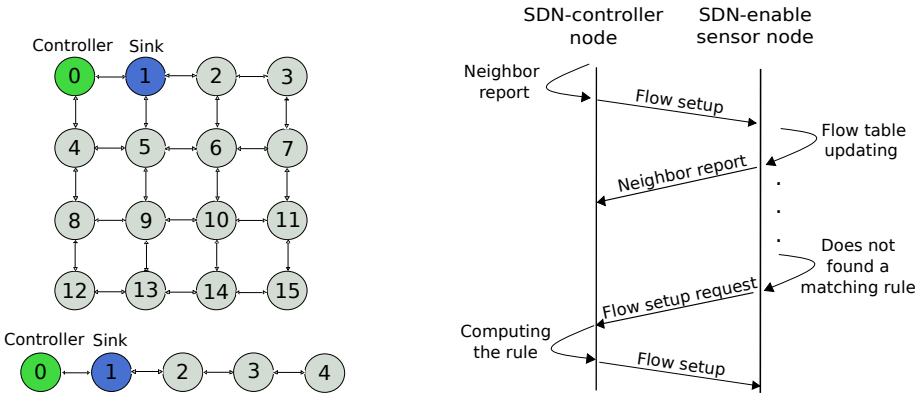
The application designed for the demonstration senses the temperature every 10 seconds and then sends its value to the sink. The demonstration of IT-SDN message exchange will occur in two phases. First we will use COOJA. For this setup we use a 4x4 grid topology, which includes one controller, one sink, and fourteen sensing nodes, as shown in Figure 7(a). Figure 7(b) depicts the message exchange, which can be visualized in the COOJA mote output. Once the controller knows its neighbors (event identified by a Neighbor Report), it sends flow setup messages, that cause the nodes to update their flow table and reply with a neighbor report. Next, if a node receives a packet for which it does not have a matching rule, it sends a flow setup request to the controller, which in turn processes the request and replies with a flow setup.

Lastly, we will execute the application developed in a small chain topology composed of 3 sensing nodes, 1 sink and 1 controller, as shown in Figure 7(a). We use the LEDs to indicate the node current state. Sensing nodes have 4 states: (1) all LEDs are off when the node does not have a route to controller; (2) red LED on indicates the node has a route to controller; (3) green LED on indicates the node has the data flow route; and (4) blue LED indicates the node is sending a Neighbor Report to the controller. For the sink, each LED is associated with a sensing node, i.e., it toggles the corresponding LED when the sink receives a data packet.

5. Final remarks

This paper presented IT-SDN, an SDN-based WSN framework. Its main features include OS-independent specification, clearly defined boundaries of underlying neighbor discovery protocol,

[2] `www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf`



(a) Topologies used in COOJA (top) and testbed (bottom)

(b) Messages exchange diagram

Figure 7. Topologies and messages exchange diagram for demonstration

support to source routed and multiple flow setups and statistics gathering. Future work includes integrating security mechanisms, including more complex flow table actions and enhancing controller decision mechanisms.

Acknowledgements

Authors would like to thank Filipe Calasans for helping with serial communication.
C. B. Margi is supported by CNPq research fellowship #307304/2015-9. R. C. A. Alves is supported by CNPq PhD fellowship #155372/2016-5. Gutavo N. Segura is supported by CAPES-PROEX fellowship #0212083 and the University of Costa Rica.

References

Chen, L. and Bian, K. (2016). Neighbor discovery in mobile sensing applications: A comprehensive survey. *Ad Hoc Networks*, 48:38–52.

de Oliveira, B. T., Margi, C. B., and Gabriel, L. B. (2014). TinySDN: Enabling multiple controllers for software-defined wireless sensor networks. In *IEEE LATINCOM*.

Dunkels, A., Gronvall, B., and Voigt, T. (2004). Contiki - a lightweight and flexible operating system for tiny networked sensors. In *IEEE LCN*, pages 455–462.

Galluccio, L., Milardo, S., Morabito, G., and Palazzo, S. (2015). SDN-WISE: Design, prototyping and experimentation of a stateful sdn solution for wireless sensor networks. In *IEEE INFOCOM*, pages 513–521.

Kobo, H. I., Mahfouz, A. M., and Hancke, G. P. (2017). A survey on software-defined wireless sensor networks: Challenges and design requirements. *IEEE Access*.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., and Voigt, T. (2006). Cross-level sensor network simulation with cooja. In *IEEE LCN*, pages 641–648.

SDNVoIP: gerenciamento de recursos de serviços de Voz sobre IP baseado em Redes Definidas por Software

Paulo Roberto Vieira Jr [1], Anderson H. S. Marcondes[1], Guilherme P. Koslovski[1], Adriano Fiorese [1]

[1]Programa de Pós-Graduação em Computação Aplicada (PPGCA)
Universidade do Estado de Santa Catarina (UDESC) — Joinville, SC — Brasil

paulorvj@gmail.com, {adriano.fiorese, guilherme.koslovski}@udesc.br
anderson.marcondes@sfs.ifc.edu.br

1. Introdução

Este trabalho apresenta o módulo SDNVoIP, uma aplicação para controle de tráfego VoIP em SDN. A aplicação gerencia o tráfego VoIP, reduzindo a utilização de CPU em servidores e a largura de banda utilizada no segmento do servidor VoIP em redes SDN. Para reduzir o uso de CPU nos servidores, SDNVoIP reconfigura as chamadas que utilizem o mesmo *Codec* para realizarem uma comunicação fim-a-fim, sem passar pelo servidor. Ou seja, o cenário da Figura 1(a) é transformado no cenário (b), sem a necessidade reconfigurar os servidores VoIP.

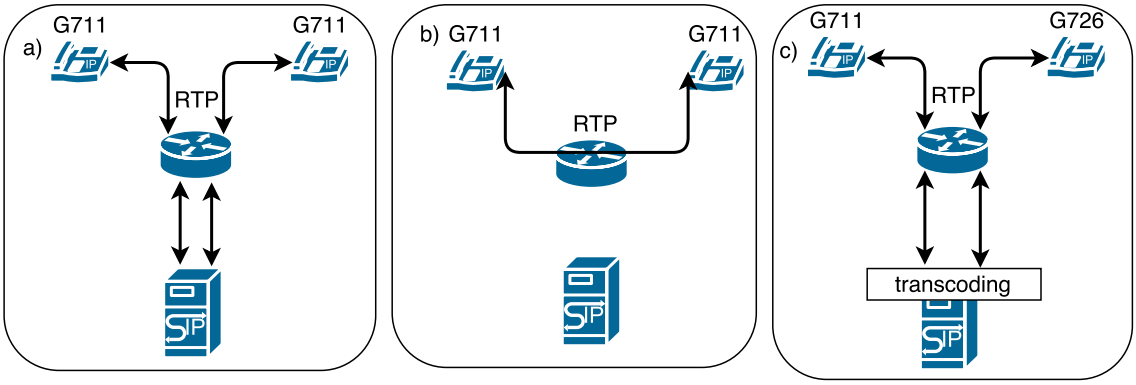


Figura 1. Exemplos de tráfego VoIP com *Codec* único (a e b) e com *Codecs* distintos (c).

A Figura 1 apresenta três maneiras pelas quais pode ocorrer o tráfego de dados em chamadas VoIP: (a) Os dados de uma chamada com o mesmo *Codec* de áudio (G711, por exemplo) são trafegados de um cliente para o outro através do servidor VoIP. Nesse caso o servidor está no caminho do áudio e dois canais são criados, um para cada cliente. O servidor VoIP lê o áudio de um canal e escreve no outro, aumentando o uso de memória e de CPU durante a leitura e escrita dos canais. Essa configuração é comumente utilizada para monitorar chamadas, coletando informações relacionadas com a qualidade do atendimento prestado por centrais de vendas. (b) O tráfego de sinalização de clientes com o mesmo *Codec* de áudio e os dados de voz são trafegados diretamente entre os clientes. Nesse caso, o servidor VoIP não está no caminho do áudio e não é possível gravar a chamada, acrescentar música de fundo, transferir ou colocar a chamada em espera. Entretanto, essa abordagem reduz o uso de CPU, memória alocada e largura de banda utilizada

pelo servidor. (c) Clientes com *Codecs* diferentes e os dados são trafegados sempre pelo servidor VoIP. Semelhante ao cenário (a), o servidor está no caminho do áudio e dois canais são criados, um para cada cliente. Entretanto, o servidor VoIP efetua a tradução dos pacotes de voz de acordo com os *Codecs* utilizados para que os clientes possam receber e ouvir o áudio [Goode 2002], aumentando o consumo dos recursos do servidor.

2. Requisitos de software

Para instalar e utilizar o módulo SDNVoIP é necessário os seguintes *softwares*:

- Java 1.7 ou posterior
- Eclipse 4.5 ou posterior
- Git

O código fonte pode ser encontrado em:

`https://github.com/paulorvj/sdnvoip`

O projeto foi desenvolvido como um módulo do controlador Floodlight [Project Floodlight 2016] e o código disponível no *link* acima possui um projeto completo e configurado para ser importado na IDE Eclipse, não sendo necessário fazer a instalação isolada do controlador Floodlight. Informações sobre como instalar e construir o controlador Floodlight podem ser encontradas em:

`https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343544/Installation+Guide`

Informações sobre como desenvolver um módulo para o Floodlight podem ser encontradas em:

`https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343513/How+to+Write+a+Module`

2.1. Importando o projeto

Primeiro é necessário fazer um clone do repositório Git, para isso utilize o comando (Figura 2):

git clone https://github.com/paulorvj/sdnvoip.git

```
wy63xzy-2:sdnvoip paulorvj$ git clone https://github.com/paulorvj/sdnvoip.git
Cloning into 'sdnvoip'...
remote: Counting objects: 2917, done.
remote: Compressing objects: 100% (896/896), done.
remote: Total 2917 (delta 1971), reused 2917 (delta 1971), pack-reused 0
Receiving objects: 100% (2917/2917), 53.06 MiB | 2.38 MiB/s, done.
Resolving deltas: 100% (1971/1971), done.
wy63xzy-2:sdnvoip paulorvj$
```

Figura 2. Git clone.

Após fazer o clone, abra o Eclipse e vá em “File - Import - Existing projects into workspace” (Figura 3):

Selecione o diretório onde foi realizado o clone do repositório, marque o nome do projeto e clique em “Finish” (Figura 4):

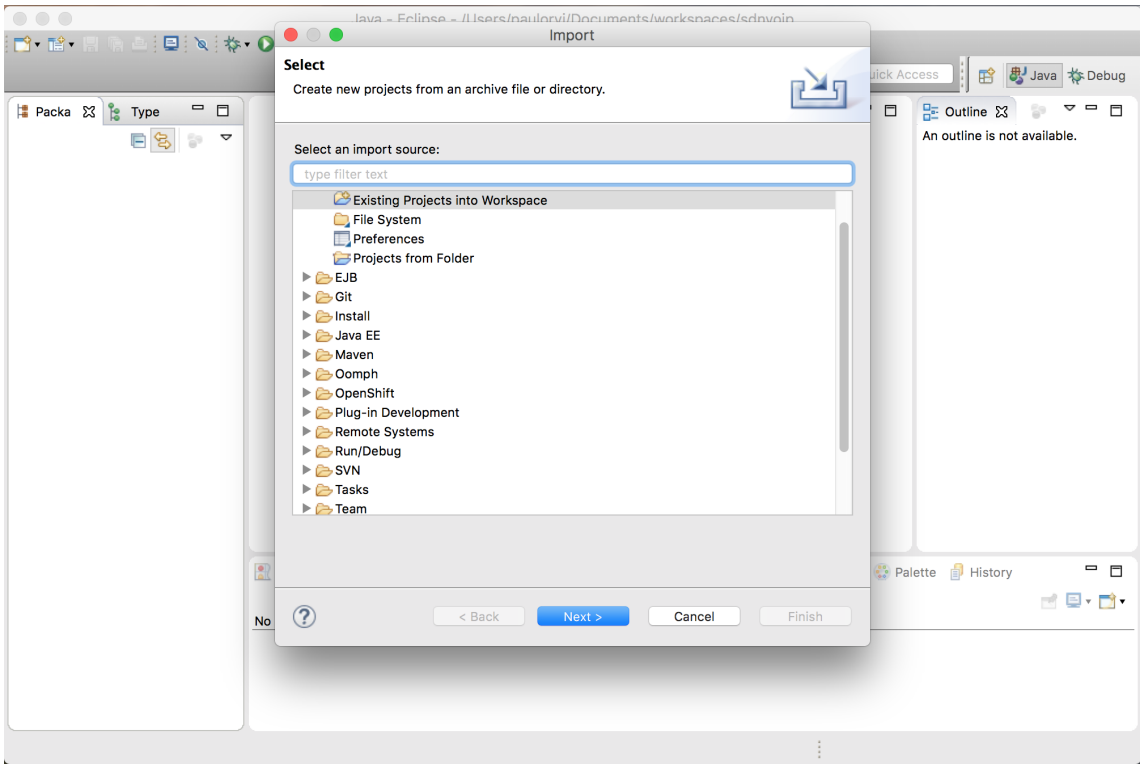


Figura 3. Importando o projeto no eclipse.

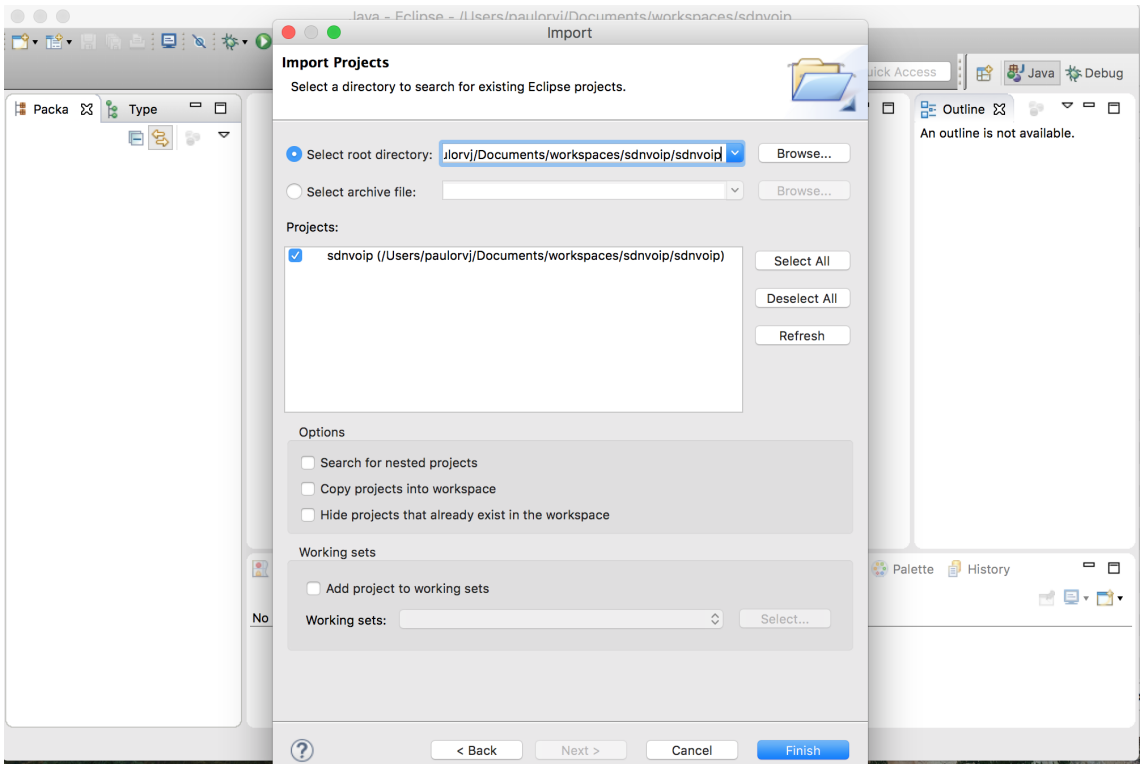


Figura 4. Importando o projeto no eclipse.

Após a importação o projeto estará pronto para execução no Eclipse. Para executar

o controlador clique em “Run - Run configurations”, clique em “Java Application” e depois no botão “New launch configuration” e em “Main class” digite (Figura 5):

net.floodlightcontroller.core.Main

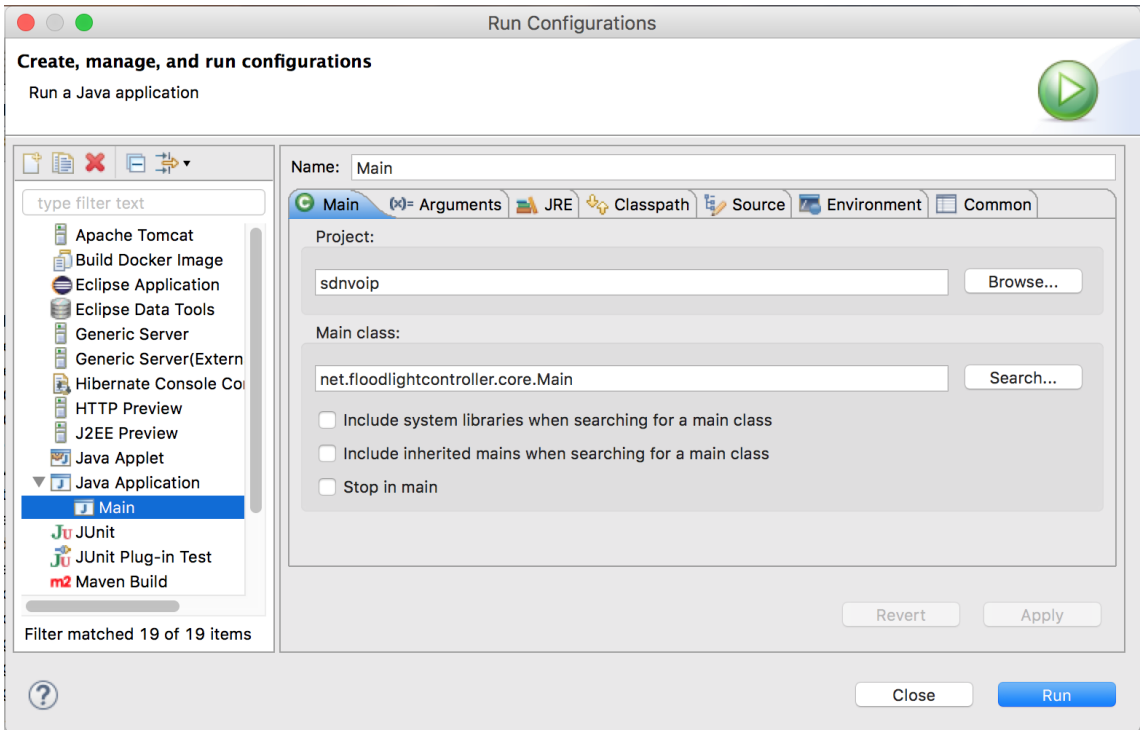


Figura 5. Run configuration.

Clique em “Run” e caso não aconteça nenhum erro o controlador será executado (Figura 6):

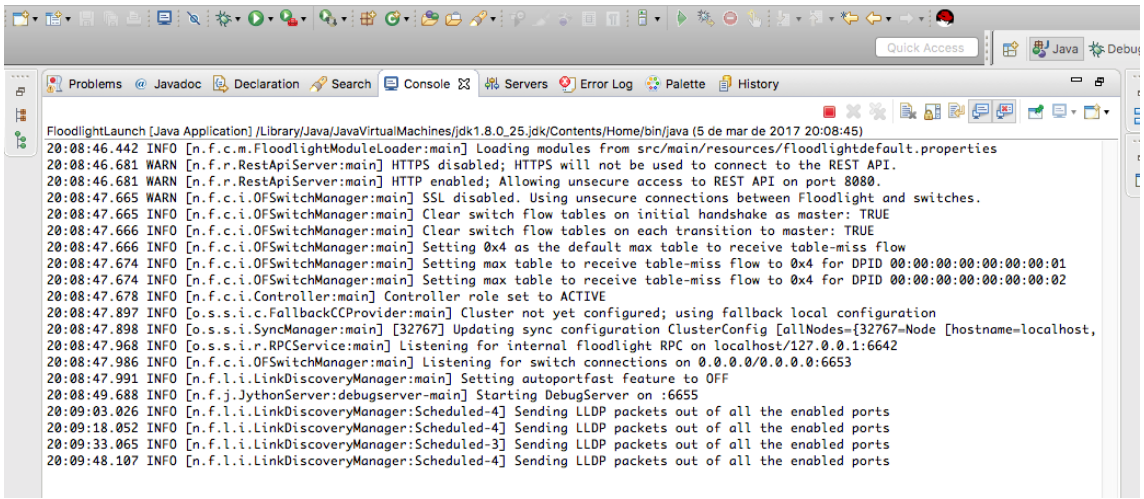


Figura 6. Execução do controlador

A partir desse momento o controlador está pronto para detectar chamadas VoIP que utilizem o mesmo Codec, não é necessário nenhuma configuração adicional.

2.2. Arquivos importantes

O código fonte do módulo encontra-se no seguinte caminho:

/src/main/java/net/floodlightcontroller/voip/Voip.java

Para que o módulo funcione corretamente é necessário que os IPs dos servidores VoIP sejam colocados em um arquivo texto, o arquivo está localizado no seguinte caminho e deve conter um IP por linha:

/sdnvoip/src/main/resources/voipservers.txt

O módulo SDNVoIP mantém informações dos clientes que estão executando chamadas, permitindo a instalação dos fluxos nos *switches* OpenFlow no caminho entre eles.

Quatro fluxos por chamada são instalados nos *switches* SDN. Dois fluxos para o caminho no sentido origem-destino e destino-origem dos pacotes RTP; e dois fluxos para o caminho no sentido origem-destino e destino-origem dos pacotes RTCP. Para cada fluxo VoIP, sete ações [Open Networking Foundation 2012] são aplicadas nos pacotes RTP e RTCP que trafegam no *switch* correspondente:

(i) Alterar o endereço MAC de origem para o endereço MAC do servidor VoIP;
(ii) Alterar o endereço MAC de destino para o endereço MAC do cliente destino;
(iii) Alterar o endereço IP de origem para o endereço IP do servidor VoIP;
(iv) Alterar o endereço IP de destino para o endereço IP do cliente destino;
(v) Alterar a porta RTP/RTCP de origem para a porta RTP/RTCP do servidor VoIP;
(vi) Alterar a porta RTP/RTCP de destino para a porta RTP/RTCP do cliente destino;
(vii) Saída do pacote em uma porta física do *switch*.

As entradas nas tabelas de fluxos devem ser instaladas nos *switches* que compõem o caminho da chamada com um tempo de expiração. Quando o término de uma chamada é sinalizado pelos clientes, a entrada é removida. Em caso de interrupção abrupta da chamada, as entradas relacionadas são automaticamente removidas dos *switches* após o tempo de expiração.

2.3. Topologia

Para demonstrar a solução em um cenário com SDN a seguinte topologia foi utilizada (Figura 7): um servidor VoIP executando Asterisk 13.01, um *switch* TP-Link WR1043 com OpenWRT e OVS habilitados, um controlador, um notebook executando *softphone* Linphone representando o cliente origem (CO) e por fim, um PC executando *softphone* Linphone representando o cliente destino (CD).

A chamada feita entre CO e CD é realizada utilizando o mesmo *Codec* (G711) e todo o áudio por padrão é trafegado passando pelo servidor VoIP.

2.4. Chamadas VoIP sem SDN

A primeira demonstração é com uma chamada entre CO e CD sem a utilização de SDN, dessa maneira todo o audio da chamada passa pelo servidor VoIP.

O servidor VoIP deve estar em execução e os dois clientes são iniciados, podemos observar que os dois clientes (u1 e u2) iniciam uma sessão no servidor VoIP (Figura 8):

Em seguida o cliente u1 incia uma chamada para o cliente u2, observa-se na Figura 9 o inicio da chamada e o tráfego RTP passando pelo servidor VoIP:

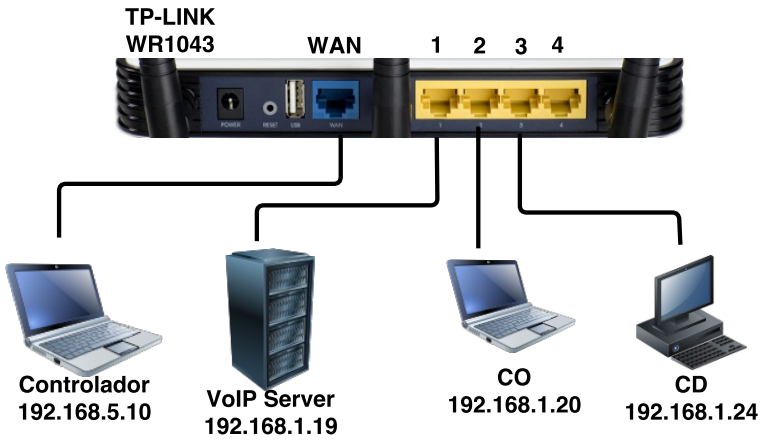


Figura 7. Cenário experimental com SDN.

```
root@paulorvj-EP43-DS3L: ~
root@paulorvj-EP43-DS3L:~# asterisk -rvvvvvvvvg
Asterisk 13.1.0~dfsg-1.1ubuntu4, Copyright (C) 1999 - 2014, Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty' for details.
This is free software, with components licensed under the GNU General Public
License version 2 and other licenses; you are welcome to redistribute it under
certain conditions. Type 'core show license' for details.
=========================================================================
Connected to Asterisk 13.1.0~dfsg-1.1ubuntu4 currently running on paulorvj-EP43-DS3L (pid = 9866)
paulorvj-EP43-DS3L*CLI>
paulorvj-EP43-DS3L*CLI>
paulorvj-EP43-DS3L*CLI>
    -- Registered SIP 'u1' at 192.168.1.20:20000
    -- Registered SIP 'u2' at 192.168.1.24:20001
```

Figura 8. Registro dos clientes no servidor VoIP

```
  == Using SIP RTP CoS mark 5
    -- Executing [u2@teste:1] Answer("SIP/u1-00000002", "") in new stack
       > 0x7fee24008730 -- Probation passed - setting RTP source address to 192.168.1.20:25000
    -- Executing [u2@teste:2] Dial("SIP/u1-00000002", "SIP/u2") in new stack
  == Using SIP RTP CoS mark 5
    -- Called SIP/u2
    -- SIP/u2-00000003 is ringing
    -- SIP/u2-00000003 answered SIP/u1-00000002
    -- Channel SIP/u1-00000002 joined 'simple_bridge' basic-bridge <35d483d0-33f6-4410-8c63-dbfd8e424018>
    -- Channel SIP/u2-00000003 joined 'simple_bridge' basic-bridge <35d483d0-33f6-4410-8c63-dbfd8e424018>
       > Bridge 35d483d0-33f6-4410-8c63-dbfd8e424018: switching from simple_bridge technology to native_r
tp
       > 0x7fee48005ef0 -- Probation passed - setting RTP source address to 192.168.1.24:25002
paulorvj-EP43-DS3L*CLI> rtp set debug on
RTP Debugging Enabled
Sent RTP P2P packet to 192.168.1.24:25002 (type 08, len 000160)
Sent RTP P2P packet to 192.168.1.20:25000 (type 08, len 000160)
Sent RTP P2P packet to 192.168.1.24:25002 (type 08, len 000160)
Sent RTP P2P packet to 192.168.1.24:25002 (type 08, len 000160)
Sent RTP P2P packet to 192.168.1.20:25000 (type 08, len 000160)
```

Figura 9. Chamada VoIP e tráfego RTP

2.5. Chamadas VoIP com SDN

Para a demonstração com SDN foi executado o controlador Floodlight e feita a reconfiguração do *switch* TP-Link WR1043 com OpenWRT e OVS habilitados e comunicando-se com o controlador. Assim como no cenário sem SDN, todo o áudio entre os clientes é trafegado pelo servidor VoIP. Entretanto, após o controlador obter todas as informações necessárias, quatro fluxos para cada chamada são instalados no *switch*, fazendo com que o áudio das chamadas não trafegue pelo servidor VoIP, fluindo diretamente entre os clientes.

Para comprovar a redução na utilização de recursos, dois cenários foram utilizados para realização dos experimentos. Um cenário sem a utilização do paradigma SDN e outro com SDN, ambos em ambiente LAN. Foi coletado a taxa de utilização de CPU utilizado o comando mpstat. O intervalo de coleta foi de 1 segundo durante o período de cinco minutos. Os dados foram tratados da seguinte maneira: para cada amostra foi calculada a média aritmética, em seguida foi calculada a variância e o desvio padrão. O servidor utilizado possui processador Intel Quad Core 3.4GHz, com 8GB de memória RAM.

A Tabela 1 apresenta os dados obtidos de utilização de CPU em um servidor VoIP com as respectivas cargas. Observa-se que 300 chamadas simultâneas utilizando o mesmo Codec e todo o tráfego passando pelo servidor VoIP, gera uma taxa de utilização de CPU de aproximadamente 27%. Já com o uso de SDN, todo o tráfego gerado pelas chamadas é encaminhado diretamente entre os clientes, fazendo com que a CPU do servidor VoIP fique aproximadamente 99% do tempo ociosa.

Tabela 1. Comparação do uso de CPU

Sem SDN						
Chamadas simultâneas	**50**	**100**	**150**	**200**	**250**	**300**
Uso de CPU%	4,03	9,22	13,92	19,35	22,78	27,92
Variância	0,53	5,59	1,33	1,04	1,37	2,74
Desv. Padrão	0,13	0,37	0,21	0,17	0,21	0,30
Com SDN						
Uso de CPU%	0,75	0,73	0,71	0,67	0,71	0,66
Variância	0,01	0,00	0,00	0,00	0,00	0,00
Desv. Padrão	0,08	0,01	0,03	0,06	0,03	0,02

Um video demonstrando o funcionamento do módulo pode ser encontrado em:

`https://www.youtube.com/watch?v=f9tir_EPIIU`

3. Considerações Finais

O presente trabalho agregou benefícios do uso de SDN na área de telecomunicações, mais especificamente na utilização de SDN para reduzir a utilização de recursos em um servidor VoIP. Os resultados obtidos mostraram que é possível reduzir a utilização de CPU em um servidor VoIP através da criação de fluxos que redirecionam o tráfego de voz diretamente entre as partes envolvidas que utilizam o mesmo Codec. Nesse caso, há também a redução na utilização da largura de banda no servidor VoIP e consequentemente redução no tráfego de dados em uma seção da rede.

A redução da utilização de CPU permite que haja maior disponibilidade de processador para chamadas que precisem de tradução de Codec, assim como a redução na largura de banda pode permitir que mais chamadas possam ser adicionadas no servidor, além de reduzir o tráfego em uma seção da rede(ex: *switch*/servidor, servidor/*switch*). Por fim, é possível redimensionar a quantidade de chamadas suportada pelos servidores VoIP existentes em uma rede, sem a necessidade de aquisição de outro servidor para suportar mais chamadas.

Referências

Chen, W.-E., Huang, Y.-L., and Chao, H.-C. (2008). Nat traversing solutions for sip applications. *EURASIP Journal on Wireless Communications and Networking*, 2008(1):1–9.

Goode, B. (2002). Voice over internet protocol (voip). *Proceedings of the IEEE*, 90(9):1495–1517.

Handley, M. (2006). Why the internet only just works. *BT Technology Journal*, 24(3):119–129.

Jain, R. and Paul, S. (2013). Network virtualization and software defined networking for cloud computing: A survey. *IEEE Communications Magazine*, 51(11):24–31.

Karapantazis, S. and Pavlidou, F. N. (2009). VoIP: A comprehensive survey on a promising technology. *Computer Networks*, 53(12):2050–2090.

Khirul, I., Islam, M. K., and Hasan, K. A. (2007). An efficient approach for nat traversal problem on security of voice over internet protocol. In *10th International Conference on Computer and information technology, 2007*, pages 1–5. IEEE.

Khlifi, H., Gregoire, J., and Phillips, J. (2006). Voip and nat/firewalls: issues, traversal techniques, and a real-world solution. *IEEE Communications Magazine*, 44(7):93.

Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Lin, Y.-D., Tseng, C.-C., Ho, C.-Y., and Wu, Y.-H. (2010). How nat-compatible are voip applications? *IEEE Communications Magazine*, 48(12):58–65.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

Open Networking Foundation (2012). OpenFlow Switch Specification. `http://www.cs.yale.edu/homes/yu-minlan/teach/csci599-fall12/papers/openflow-spec-v1.3.0.pdf`.

Park, C., Jeong, K., Kim, S., and Lee, Y. (2008). Nat issues in the remote management of home network devices. *IEEE network*, 22(5):48–55.

Project Floodlight (2016). Project floodlight: Open source software for building software-defined networks. `http://www.projectfloodlight.org/floodlight`.

Yeryomin, Y., Evers, F., and Seitz, J. (2008). Solving the firewall and nat traversal issues for sip-based voip. In *Telecommunications (ICT), International Conference on*, pages 1–6. IEEE.

SNMP Gateway CCN: Software de gerência de redes orientadas a conteúdo interoperável com sistemas legados

Marciel de Lima Oliveira[1], **Christian Esteve Rothenberg**[1]

[1] Departamento de Engenharia da Computação e Automação Industrial (DCA)
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (UNICAMP)
Av Albert Einstein, 400, Cidade Universitária Zeferino Vaz, Campinas, SP, Brasil

{marciel,chesteve}@dca.fee.unicamp.br

Abstract. *Research on Information-Centric Networking (ICN) mainly focuses on the data and control plane design and implementation challenges compared to the efforts devoted on the management plane. Aiming at addressing this gap, this paper presents some mapping mechanisms and the SNMP Gateway CCN tool, enabling the management and monitoring of CCN nodes through legacy SNMP-based systems. The tool allows exploring the concept of content-oriented management (name/data) in emulated environments.*

Resumo. *Pesquisas voltadas as Redes Orientadas a Conteúdo (ROCs) tem maior foco nos planos de dados e controle em comparação ao plano de gerência. Como contribuição para suprir essa carência, este artigo apresenta alguns mecanismos de mapeamento e a ferramenta SNMP Gateway CCN, que permite o gerenciamento e monitoramento de nós CCN através de sistemas de gerência de redes legadas baseados no protocolo SNMP (Simple Network Management Protocol). A ferramenta possibilita explorar o conceito de gerência orientada a conteúdo (nomes/dados) em ambientes emulados.*

1. Introdução

Com o surgimento de grandes redes de transporte e equipamentos complexos construídos para tratar diversos serviços como dados, voz e vídeo, surge também o interesse de monitorar e otimizar o uso destas redes (equipamentos e serviços). Esse monitoramento é classificado como *plano de gerência* como forma de diferenciá-lo dos *planos de dados e de controle*, responsáveis pela implementação efetiva dos serviços oferecidos aos usuários. Parte da instanciação do plano de gerência se dá através da ideia do centro de operações de redes (NOC) que atua em um regime 24/7 para a operação, manutenção e análise do desempenho das redes e dos respectivos equipamentos.

O plano de gerência geralmente conta com um sistema de gerência de redes central NMS (*Network Management System*), que atua no monitoramento e operação das três grandes divisões das redes (plano de dados) de equipamentos de telecomunicações, núcleo (ex.: DWDM), agregação (ex.: MPSL-TP, Metro-Ethernet) e acesso (ex.: LTE/4G, xDSL, FTTH). A comunicação entre os sistemas de gerência e os equipamentos no plano de dados é feita através de uma rede dedicada chamada DCN (*Data Communication Network*) e os protocolos de rede TCP/IP são adotados como padrão para uso nos equipamentos que compõem a DCN (roteadores L3/IP/MPLS e switches L2/Ethernet/Metro).

O surgimento de recentes trabalhos em Redes Orientadas a Conteúdo (ROCs) representa um novo paradigma onde o foco das redes é baseado no conteúdo e não mais na sua localização [de Brito et al. 2012]. As ROCs propõem que o conteúdo seja o elemento central das redes, independente de sua localização, substituindo o foco *de onde* para *o quê*. Nas ROCs, a infraestrutura da rede participa ativamente no armazenamento (*caching*) e na distribuição dos conteúdos visando um aumento na eficiência da busca e na disponibilidade dos conteúdos na rede.

As ROCs têm despertado grande interesse no meio acadêmico e dentre várias empresas e institutos relacionados às pesquisas na área das novas arquiteturas de rede abrindo espaço para novas aplicações, pesquisas e experimentos, tais como: CCN [Jacobson et al. 2009], que apresenta uma estrutura hierárquica para nomes semelhante às URLs; DONA [Koponen et al. 2007], que utiliza o mecanismo de nomeação plana e funções de hash criptográfico e LIPSIN [Jokela et al. 2009] que possui uma arquitetura que identifica os enlaces pelo nome ao invés dos pares de endereços fim a fim.

O novo paradigma proposto pelas ROCs traz consigo inúmeros desafios [Xylomenos et al. 2014]. Esse trabalho foca no ponto de vista de gerência de redes orientadas a conteúdo, levando em consideração a carência, tanto no nível de mecanismos adequados, como na definição de um plano de gerência para estas redes.

2. Motivação e Objetivo

A motivação principal deste artigo deve-se à percepção da carência de paradigmas adequados à gerência de redes orientadas a conteúdo [Kutscher et al. 2016]). Consideramos a possibilidade de experimentar gerência de redes orientadas a conteúdo com o uso de protocolos e arquiteturas de gerência utilizados nas redes tradicionais, como por exemplo; TL1, REST, SNMP, CLI , WEB UI e NETCONF/YANG [web 2014, Nunes and David 2005, James and ROSS 2006, Mauro and Schmidt 2005, net 2014, Schonwalder et al. 2010], transformando-as em ferramentas eficientes para a gerência de redes CCN [Oliveira and Rothenberg 2014].

A arquitetura CCN (*Content-Centric Networking*) [Jacobson et al. 2009] adotada como referência nesse trabalho é reconhecidamente uma das propostas mais relevantes na literatura relativamente às redes orientadas a conteúdo. As redes CCN utilizam uma estrutura de nomes hierárquicos e legíveis (formados por sequências de caracteres e números) para identificar os conteúdos. Tais nomes possuem características semânticas, ou seja, os componentes hierárquicos utilizados na identificação trazem algum tipo de informação sobre o conteúdo, como por exemplo, versão, formato ou propriedade.

Para tornar os sistemas de gerência compatíveis, esta proposta define como estratégia a utilização do label/nome como identificador único de um nó na rede CCN e o mapeamento de mensagens através de um gateway. Essa estratégia é denominada NONM (*Named-Oriented Network Management*) [Oliveira and Rothenberg 2014], tem como principal tarefa compatibilizar a gerência tradicional baseada no protocolo SNMP (*Simple Network Management Protocol*), com o modelo CCN, dessa forma permitir a gerência de elementos de rede CCN nativos.

3. Projeto e arquitetura para gerência de redes orientada a conteúdo

A modelagem de uma ferramenta SNMP Gateway CCN é o primeiro passo em direção à adoção de mecanismos para gerência de redes CCN, sejam nativas ou overlay. Para gerenciar elementos das redes orientadas a conteúdo, optamos pelo SNMP por se tratar de um protocolo largamente conhecido, que se mostra mais eficiente do ponto de vista de monitoramento e alarmes, em relação a outros protocolos que tem maior destaque na configuração. A ferramenta SNMP Gateway CCN tem como uma de suas principais características o uso de uma arquitetura de gerência de redes *in-band*, que atua em conjunto com o plano de dados para monitoramento dos elementos de rede. Neste contexto o pacote de interesse (requisição) é formado pelo nome do conteúdo monitorado que deseja consultar e o *payload* do pacote de dados (resposta) carrega o valor do objeto consultado.

3.1. Mecanismo de tradução SCNT

O método SCNT (*SNMP Content Network Translation*) tem como principal objetivo permitir a utilização do campo *ContextName* (string formada em texto plano) da mensagem PDU do protocolo SNMPv3, como meio para passagem de parâmetro para informar ao gateway qual elemento deseja consultar na rede CCN. O campo *Community* existente nas versões SNMPv1 e SNMPv2 do protocolo também poderia ser usado para esse propósito, porém a funcionalidade do campo teria que ser redefinida. Por tanto, a ferramenta *SNMP Gateway CCN* não possui suporte para as versões 1 e 2 do protocolo. O valor do campo *Object-Name* que representa o OID (*Object identifier*), é utilizado em conjunto com o valor do campo *contextName*, para formar a mensagem *Interest* no processo de consulta. Após a entrega da mensagem *Interest* do *SNMP Gateway CCN* para a rede CCN, formada pelo mapeamento descrito anteriormente, a rede deve retornar como resposta um pacote *Data*, levando em consideração a arquitetura do modelo CCN. Quando o *SNMP Gateway CCN* recebe o pacote *Data* de volta como resposta, a mensagem *PDU Response* é formada de acordo com o conteúdo recebido e é encaminhada de volta para o Gerente (NMS) que fez a solicitação no início no processo.

3.2. MIB CCN

Neste trabalho definimos uma MIB (*Management Information Base*) [Walsh 2008] para a rede CCN com o mesmo propósito da MIB apresentada em [Kang et al. 2012], que se diferencia nos aspectos relativos à gerência de objetos refletidos em um agente CCN nativo, ao contrário de um agente SNMP convencional, e posicionamento sob o ramo da MIB-2 com OID 100, ao invés do ramo *private*, desta forma podemos utilizar a filosofia de extensibilidade da mib padrão. A MIB CCN é apresentada de forma resumida na Figura 1.

A MIB CCN tem como objetivo criar novos ramos na árvore com a identificação de objetos exclusivos (OIDs) para monitoramento de elementos de rede CCN em redes nativas. A MIB CCN proposta fica sob hierarquia da MIB-2 e está classificada em duas partes, a primeira parte representa objetos de *ccnSystem* que tratam informações do próprio nó, e a segunda parte trata objetos específicos para monitoramento de protocolo CCN através da coleta de dados da daemon *ccndStatus*. O nó CCN possui características que o torna único em relação à arquitetura de outros elementos das redes, pois possui tabelas de controle diferenciadas para tratamento e roteamento de pacotes/conteúdo.

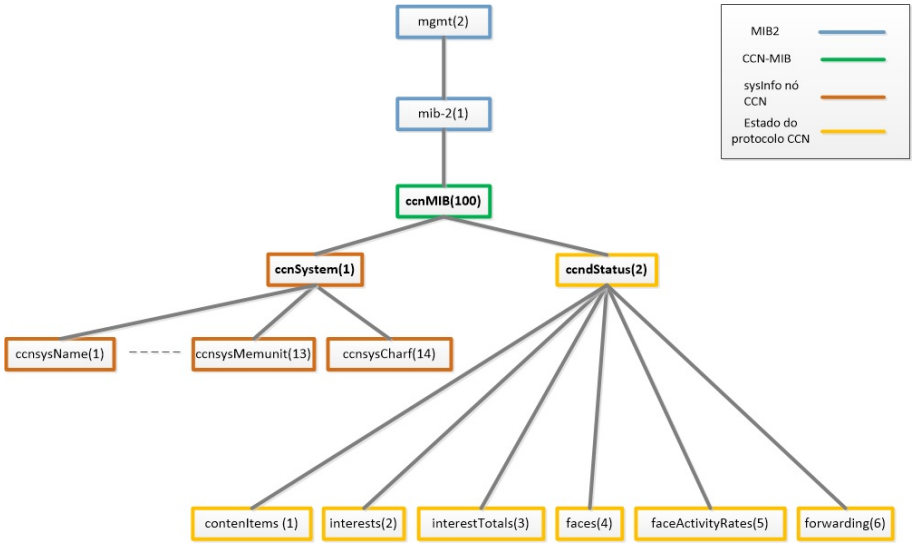


Figura 1. A MIB CCN e sua sub-árvore.

3.3. Visão geral da ferramenta

A ferramenta *SNMP Gateway CCN* foi implementada tendo como foco a operação *GET*, levando em consideração as garantias das funcionalidades mínimas. A plataforma *MiniCCNx* [Cabral et al. 2013] foi utilizada como ambiente para troca de mensagens entre o *gerente IP* e o *Agente CCN*, que também compõe a rede de elementos CCN. A plataforma *MiniCCNx* foi escolhida por ser considerada robusta e confiável para coleta de evidências que comprovem o funcionamento e eficácia dos experimentos. Do ponto de vista de um ambiente CCN nativo, a plataforma *MiniCCNx* possui características de um emulador que permite executar aplicações reais, necessárias para auxiliar na interação do *SNMP Gateway CCN*, com o agente nativo do ambiente emulado.

A ferramenta *SNMP Gateway CCN* é composta basicamente das partes; *MIB CCN*, *Agente SNMP* e as ferramentas *ccnmanager* e *ccnagent* (Agente CCN nativo) que compõem a arquitetura *SCNT* .

MIB CCN. A MIB CCN deve implementar todos os objetos (OIDs) definidos para gerência e monitoramento dos elementos CCN nativos. A implementação da MIB segue as especificações da RFC 3418, definida com base no padrão SMI (*Structure of Management Information*).
Agente SNMP. O Agente SNMP é capaz de interpretar as consultas SNMP feitas para o objetos mapeados na MIB CCN da hierarquia, em seguida traduz as mensagens que são encaminhadas para rede de elementos CCN nativos.
Agente CCN. O agente CCN nativo, tem como principal objetivo fornecer conteúdo em resposta às requisições originadas do gateway, os conteúdos devem refletir exatamente cada objeto do elemento de rede gerenciado, que estão diretamente relacionados com os OIDs especificados na MIB CCN.

Conforme ilustrado na Figura 2, a ferramenta *SNMP Gateway CCN* é capaz de receber mensagens SNMP do tipo *Request* e traduzi-las para mensagens CCN do tipo

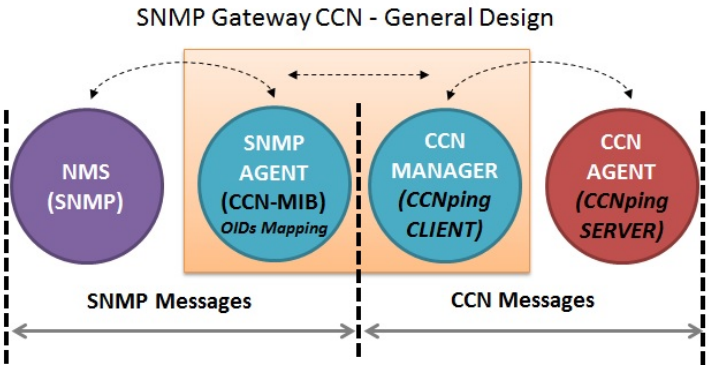


Figura 2. Visão geral da ferramenta *SNMP Gateway CCN*

Interest, o mesmo processo deve ocorrer no sentido inverso, ou seja, receber mensagens CCN do tipo *Data* e convertê-las para mensagens SNMP do tipo *Response*, o mecanismo de tradução atua de forma totalmente transparente do ponto de vista do NMS.

4. Plano de demonstração da ferramenta

Os principais componentes da ferramenta serão apresentados inicialmente, assim como as configurações necessárias para preparação do ambiente emulado. O MIB Browser *SnmpB* deve ser utilizado para carregar a MIB CCN que por sua vez será responsável por gerar as mensagens de consulta SNMP encaminhadas ao Agente SNMP executado no gateway, por tanto, terá o papel do NMS (vide Figura 3 ilustrativa).

A ferramenta será demonstrada em duas etapas, na primeira etapa o funcionamento da operação *SNMP GET* será avaliado passo a passo, na segunda etapa devemos exercitar o funcionamento da operação *SNMP WALK*, que tem o objetivo de consultar todos os objetos da MIB CCN.

A topologia de referência adotada para validação da prova de conceito, apresenta um cenário com 10 elementos de rede CCN, pré-configurado no ambiente MiniCCNx. Com a auxílio da ferramenta MiniccnxEdit, a topologia foi construída em anel, com alguns elementos ramificados no formato linear. Por fim, todas as configurações da topologia foram armazenadas no arquivo ***r1-r10.mnccnx*** para facilitar a restauração do ambiente quando necessário. Para nomeação dos elementos de rede, adotamos o prefixo ***r*** que representa a inicial da palavra ***router***, seguido de um número de 1 até 10.

Para verificar a troca de mensagens SNMP e CCN, adotamos o analisador de protocolos de rede *Wireshark* juntamente com um plugin[1] específico para suporte à análise de mensagens do protocolo CCNx.

4.1. Teste funcional da operação SNMP GET

Tendo como base a topologia de referência, o elemento de rede CCN ***r1*** foi escolhido como gateway de rede. O objetivo inicial é consultar o valor de um objeto qualquer em um elemento de rede distante do gateway, deste modo, uma mensagem de consulta *SNMP*

[1]O plugin de CCNx para *Wireshark* foi compilado e instalado conforme instruções do projeto abaixo. https://github.com/ProjectCCNx/ccnx/blob/master/apps/wireshark/README-wireshark-1.6.txt

GET Request deve partir do elemento ***r1*** e alcançar o elemento alvo, uma mensagem *SNMP GET Response* deve retornar ao gateway em resposta à requisição. A interface de rede ***lo*** (loopback/IP 127.0.0.1) será monitorada no elemento de rede gateway para captura das mensagens SNMP. As interfaces ***eth-0*** e ***eth1*** serão monitoradas para captura das mensagens CCN. Abaixo seguem os passos e o comportamento esperado como resultado para o teste funcional da operação SNMP GET:

Passos:
- O *Mib Browser SnmpB* deve ser configurado para preencher o campo *contextName* do protocolo SNMPv3 conforme o nome do elemento de rede que deseja consultar.
- Executar o *Wireshark* para monitorar as interfaces de rede ***lo***, ***eth0*** e ***eth1*** do elemento gateway ***r1***.
- A partir do *Mib Browser*, iniciar uma consulta *SNMP GET Request* para uma OID da MIB CCN.
- Na captura realizada na interface *lo* do gateway r1, deve ser possível verificar a existência de uma mensagem de requisição *SNMP GET* na versão 3, o campo *contextName* preenchido com o label definido no *Mib Browser*, e na sequência uma mensagem de resposta *SNMP GET Response* com o valor solicitado.
- Nas capturas realizadas nas interfaces *eth0* e *eth1* do gateway, deve ser possível verificar a existência de uma mensagem CCN de requisição *Interest* e na sequência uma mensagem de resposta *Data* com o valor solicitado.

4.2. Teste funcional da operação SNMP WALK

O principal objetivo deste experimento é percorrer todos os 280 objetos da MIB CCN. Mensagens *Interest* de requisição devem partir do elemento ***r1*** e alcançar os elementos alvo, uma mensagem de resposta *Data* deve retornar ao gateway em resposta à cada requisição. As interfaces ***eth-0*** e ***eth1*** serão monitoradas para captura das mensagens CCN. Abaixo seguem os passos e o comportamento esperado como resultado para o teste funcional da operação SNMP WALK:

Passos:
- O *Mib Browser SnmpB* deve ser configurado com o valor do campo *contextName* do elemento de rede que deseja consultar, podemos exercitar a consulta para qualquer um dos elementos da topologia de referência.
- Executar o *Wireshark* para monitorar apenas as interfaces de rede ***eth0 eth1*** do elemento gateway ***r1***.
- A partir do *Mib Browser*, iniciar uma consulta *SNMP WALK Request* a partir do OID principal da MIB CCN.
- Nas capturas realizadas nas interfaces *eth0* e *eth1* do gateway, deve ser possível verificar a existência de uma mensagem CCN de requisição *Interest*, e na sequência uma mensagem de resposta *Data* para cada valor solicitado.

O código fonte aberto da ferramenta *SNMP Gateway CCN* está disponível online[2]

[2]https://github.com/marcieloliveira/snmp-gateway-ccn

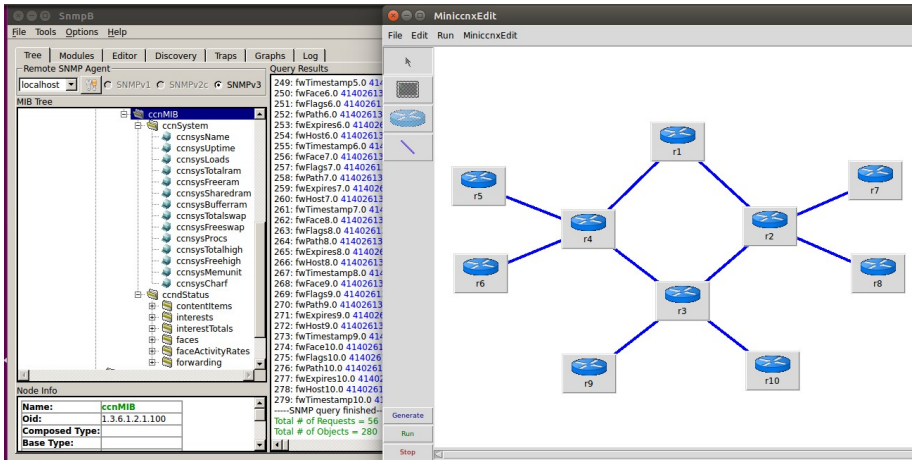


Figura 3. Ferramenta *SNMP Gateway CCN* em funcionamento.

juntamente com um tutorial para utilização, uma imagem da maquina virtual do ambiente completo e um vídeo[3] com a demonstração funcional da ferramenta.

5. Conclusão e Trabalhos Futuros

Uma solução eficiente para gerência de elementos CCN nativos com uso de gerentes SNMP em redes IP, abre espaço para novas discussões sobre a interoperabilidade entre redes legadas e redes orientadas a conteúdo.

Entre as vantagens da arquitetura proposta, é apresentada a possibilidade de tornar os sistemas convencionais (SNMP, NETCONF/YANG ou outros) compatíveis com novos sistemas e paradigmas (ex.: CCN), uma vez que a migração destas arquiteturas pode ser feita de forma gradual sem a necessidade de grandes alterações na infraestrutura já existente. Desta forma podemos afirmar que a ferramenta *SNMP Gateway CCN* serve como um facilitador no processo de migração das plataformas legadas. Outro ponto a se observar, é a possível diminuição de custos de operação das redes de telecomunicações, uma vez que o elemento gerenciado passa a ser localizado através do seu *label/nome* que se mostra totalmente desacoplado do endereçamento IP convencional, no que se refere ao controle destes endereços em grandes redes (centenas e milhares de elementos) devido a sua arquitetura mais complexa de planejamento.

A ferramenta *SNMP Gateway CCN* atualmente se limita a algumas funcionalidades, dentre elas, as operações básicas do protocolo SNMP (*GET*, *GET-NEXT*, *GET-BULK* e *WALK*) e apresenta uma topologia de referência estática com apenas 10 elementos de rede, que tem como objetivo principal, experimentar a gerência de redes CCN e demonstrar a prova de conceito. Tais limitações demonstram lacunas a serem ocupadas do ponto de vista funcional, como por exemplo, a operação *SET* do SNMP. Deste modo, novas pesquisas também podem ser conduzidas para a criação de um *Gerente CCN* nativo além do *Agente CCN* já proposto neste trabalho, que permitiria desacoplar o ambiente legado (NMS SNMP/IP) do ambiente CCN e experimentar a relação *gerente/agente* em um cenário baseado apenas no modelo CCN.

[3]https://youtu.be/vIfCsDhPoS0

References

(2014). *Netconf Central - Network Configuration Protocol*. URL: www.netconfcentral.org.

(2014). *Webnms - Introdução ao protocolo TL1*. URL: www.webnms.com.

Cabral, C., Rothenberg, C. E., and Magalhães, M. F. (2013). Mini-ccnx: Fast prototyping for named data networking. In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, pages 33–34. ACM.

de Brito, G. M., Velloso, P. B., and Moraes, I. M. (2012). Redes orientadas a conteúdo: Um novo paradigma para a internet. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC*, 2012:211–264.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009). Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM.

James, F. and ROSS, K. W. (2006). Redes de computadores e a internet: uma abordagem topdown.

Jokela, P., Zahemszky, A., Esteve Rothenberg, C., Arianfar, S., and Nikander, P. (2009). Lipsin: Line speed publish/subscribe inter-networking. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, pages 195–206, New York, NY, USA. ACM.

Kang, W., Sim, B., Kim, J., Paik, E., and Lee, Y. (2012). A network monitoring tool for ccn. In *World Telecommunications Congress (WTC), 2012*, pages 1–3. IEEE.

Koponen, T., Chawla, M., Chun, B.-G., Ermolinskiy, A., Kim, K. H., Shenker, S., and Stoica, I. (2007). A data-oriented (and beyond) network architecture. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 181–192. ACM.

Kutscher, D., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and Waehlisch, M. (2016). Information-centric networking (icn) research challenges. Technical report.

Mauro, D. and Schmidt, K. (2005). *Essential SNMP: Help for System and Network Administrators*. " O'Reilly Media, Inc.".

Nunes, S. and David, G. (2005). Uma arquitectura web para serviços web. *XATA 2005-XML: Aplicações e Tecnologias Associadas*.

Oliveira, M. and Rothenberg, C. E. (2014). Snmp proxy ccn: Uma proposta de arquitetura para gerência de redes orientadas a conteúdo interoperável com sistemas legados.

Schonwalder, J., Bjorklund, M., and Shafer, P. (2010). Network configuration management using netconf and yang. *IEEE communications magazine*, 48(9).

Walsh, L. (2008). *SNMP MIB Handbook: essential guide to MIB development, use and diagnosis*. Wyndham.

Xylomenos, G., Ververidis, C. N., Siris, V. A., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, K. V., and Polyzos, G. C. (2014). A survey of information-centric networking research. *IEEE Communications Surveys & Tutorials*, 16(2):1024–1049.

MISSIn: Orquestração Interativa de Infraestruturas SDN

Emidio P. Neto[1], **Felipe S. Dantas Silva**[1], **Kevin B. Costa**[1]
João Batista da Silva[1], **Augusto J. Venâncio Neto**[2]

[1]Laboratório de Tecnologias Avançadas em Redes de Computadores (LaTARC)
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN)
Natal-RN, Brasil

[2]Departamento de Informática e Matemática Aplicada (DIMAp)
Universidade Federal do Rio Grande do Norte (UFRN) – Natal-RN, Brasil

emidio.neto@academico.ifrn.edu.br, felipe.dantas@ifrn.edu.br
{kevin.costa,goncalves.silva}@academico.ifrn.edu.br, augusto@dimap.ufrn.br

Abstract. *The Software-Defined Networking (SDN) paradigm has introduced a set of resources with the prospect to design new services and applications tailored to the next generation of network technologies. For instance, the task of enforcing SDN features over targeted network device(s) require appropriate tools that must be comply with the available SDN controller northbound API, which is not flexible. Moreover, the remote SDN substrate access is not a trivial task, since each SDN controller approach provide specific APIs. In view of this, we introduce the MISSIn (Modular Interactive Management System for SDN Infrastructure), which allows SDN-based management ubiquitously and agnostic to the underlying SDN controller approach. Through MISSIn, a web-based interface allows enforcing OpenFlow network features in a dynamic and interactive way.*

Resumo. *O paradigma de Redes Definidas por Software (Software-Defined Networking – SDN) introduziu um conjunto de recursos com a perspectiva de projetar novos serviços e aplicações adaptadas à próxima geração de tecnologias de rede. Como exemplo, a tarefa de impor recursos SDN sobre dispositivos de rede requer ferramentas apropriadas que devem estar em conformidade com a API northbound do controlador SDN, que não é flexível. Além disso, o acesso SDN ao substrato remoto não é uma tarefa trivial, uma vez que cada abordagem de controlador SDN fornece APIs específicas. Em vista disso, apresentamos o MISSIn (Modular Interactive Management System for SDN Infrastructure), que permite que o gerenciamento baseado em SDN seja onipresente e agnóstico para a abordagem do controlador SDN subjacente. Através do MISSIn, uma interface baseada na web permite aplicar os recursos de rede OpenFlow de forma dinâmica e interativa.*

1. Introdução

Dentre as principais propostas lançadas para lidar com os problemas da atual infraestrutura da Internet, o paradigma de Redes Definidas por Software (*Software-Defined Networking* – SDN) [E. Haleplidis and Koufopavlou 2015] tem se destacado, principalmente por seu potecial de inovação e flexibilidade no desenvolvimento de novas tecnologias. Com

SDN, novos mecanismos podem ser incorporados sem a necessidade de adaptações no núcleo da rede, permitindo a prototipação de novos serviços e avaliações de novos mecanismos e estratégias de controle para aprovisionar intercomunicações com maior nível de refino (tais como Qualidade de Serviço (*Quality of Service* – QoS), balanceamento de carga, segurança, dentre outros).

Embora as facilidades proporcionadas pela tecnologia SDN maximizem a utilização das capacidades dos dispositivos de rede através de novos serviços e aplicações, alguns fatores envolvidos na gerência das infraestruturas SDN continuam a ser desafiadores em relação à integração de diferentes mecanismos [Sezer et al. 2013]. Isto aplica-se, particularmente, à falta de sistemas de controle que possam suportar os administradores de rede quando confrontados com a crescente complexidade da infraestrutura e com a vasta gama de requisitos de qualidade dos usuários [Sharma et al. 2013]. Além disso, um fator crítico comumente encontrado no gerenciamento de infraestruturas SDN é a necessidade de visualização e operação da topologia em tempo de execução, especialmente para operadores de rede com pouco ou nenhum conhecimento em desenvolvimento de software.

Os controladores SDN atualmente disponíveis exigem que o desenvolvedor de software tenha total conhecimento de sua API. Esta limitação acarreta em mau uso dos recursos de controle, maior complexidade e menor flexibilidade no gerenciamento da infraestrutura, uma vez que lida-se com cada controlador de maneira diferente em virtude da ausência de padronização. Como exemplo, supõem-se que uma rede com vários domínios administrativos, formada por vários controladores e que deseje fazer uso de uma implementação específica de uma aplicação de gerência de QoS deve ter diferentes implementações do mecanismo desejado, uma para cada controlador SDN diferente. Neste caso, todas as atualizações de software deste recurso devem ser substituídas para cada controlador individualmente. Além disso, devem ser utilizadas diferentes interfaces de gerência, o que resulta em acréscimo de complexidade no gerenciamento do ambiente.

Outro importante ponto é que a API de serviço web suportada pela maioria dos controladores SDN não oferece *feedback* em tempo real, o que significa que ela não permite que o desenvolvedor registre um *event listener* para receber mensagens de alerta em uma aplicação remota do controlador instantaneamente e sem qualquer solicitação adicional. A ausência de um mecanismo que seja capaz de lidar com essas limitações impõe uma série de restrições à tarefa de monitoramento da rede no que diz respeito ao desempenho e escalabilidade.

Uma ferramenta de orquestração SDN deve fornecer um vasto conjunto de serviços ao operador de rede para auxiliar no gerenciamento da infraestrutura como um todo e, assim, reduzir a lacuna entre a complexidade da topologia subjacente e a ampla gama de dispositivos que a compõem. Uma alternativa viável para esta demanda é a utilização de interfaces de gerência que permitam a interação entre o operador de rede e a infraestrutura, além de assumir a responsabilidade de se compatibilizar em diferentes cenários por meio da uma interface única para importação de funções de rede. Isso pode permitir a rápida identificação de novos recursos de rede e sua configuração através de modelos (*templates*) personalizáveis. Desta forma, é possível integrar novos recursos SDN através de uma interface de controle dinâmico e interativo, que isenta o operador da necessidade de ter avançados conhecimentos em desenvolvimento de software.

Neste contexto, este trabalho apresenta o *Modular Interactive Management System for SDN Infrastructure* (MISSIn), uma ferramenta de suporte para a orquestração de infraestruturas SDN que é capaz de encapsular a granularidade do plano de controle, auxiliando os operadores de rede em lidar com a complexidade envolvida nos aspectos de heterogeneidade das infraestruturas SDN, de uma forma dinâmica e interativa.

O restante deste artigo está estruturado da seguinte forma: A Seção 2 realiza um estudo de trabalhos relacionados para destacar a contribuição do MISSIn. A Seção 3 fornece uma descrição detalhada do MISSIn, incluindo sua arquitetura, componentes principais, operações e interfaces. A Seção 4 detalhalha a proposta de demonstração da ferramenta. A Seção 5 fornece algumas considerações finais e faz sugestões para trabalhos futuros.

2. Trabalhos Relacionados

Nos últimos anos tem havido uma tendência crescente, na comunidade acadêmica e na indústria, para avaliar novos conceitos por meio do paradigma SDN. No entanto, tarefas como a orquestração da infraestrutura, a visualização de eventos em tempo de execução e atualizações nos dispositivos de rede exigem conhecimento adicional relevante do controlador SDN, o que geralmente não está disponível por padrão.

Recentemente, muitos esforços tem sido dedicados à concepção de novas propostas com o objetivo de facilitar a interação entre o operador de rede e a infraestrutura de diferentes formas. Um exemplo é abstrair a complexidade de grandes blocos de código de programação para o desenvolvimento de novas aplicações e políticas operacionais através da sua integração com ferramentas de terceiros.

A literatura revela que a maioria dos esquemas se preocupam apenas em fornecer uma visualização gráfica dos dispositivos de rede. Em alguns casos, eles podem mostrar o posicionamento dos dispositivos na topologia, mas não são capazes de permitir sua interação direta, como pode ser visto em [Salsano et al. 2015] e [Huang et al. 2014]. Em [Schultz et al. 2015] os autores apresentam a proposta OpenGUFI, que consiste em um sistema com interface gráfica para abstração de infraestrutura de rede em tempo de execução. A ferramenta permite que o operador examine as conexões entre clientes móveis e pontos de acesso sem fio, com suas respectivas regras de fluxo, mas sem viabilizar ações interativas para o usuário. A proposta não aborda o problema da complexidade da rede, que ainda tem de ser gerenciada manualmente pelo administrador, não atingindo assim o objetivo de facilitar a gestão da infraestrutura.

Algumas propostas precisam ser integradas com ferramentas de extensão providas por terceiros, que por vezes podem estar disponíveis através de uma licença de uso restrito. Este é o caso do trabalho em [Shalimov et al. 2015], onde o controlador RUNOS é proposto. A proposta faz uso do mecanismo *Maple* [Voellmy et al. 2013] para tornar as aplicações mais otimizadas e adota o conceito de *multi-threading* efetivo para a modularização de processos separadamente, com o objetivo de evitar sobrecarga do sistema. No entanto, ao exigir que o operador de rede tenha um conhecimento especializado de sua API para utilizá-lo, ele vai contra a idéia de gerenciamento de rede flexível.

Outra abordagem consiste em soluções de orquestração de rede integradas com controladores SDN personalizados. Isso está sujeito a restrições graves de gerenciamento

de rede, uma vez que aplicações que foram desenvolvidas para um controlador específico só podem ser usadas por ele, impedindo a interoperabilidade entre os controladores. Algumas soluções baseadas nesta abordagem são descritas abaixo.

Em [Choi et al. 2015] os autores apresentam o IRIS-CoMan como uma solução para o gerenciamento de infraestrutura SDN em larga-escala. Ele é baseado no controlador *Floodlight* e requer a migração de todas as aplicações já desenvolvidas baseados em sua API.

Kandoo [Hassas Yeganeh and Ganjali 2012] organiza os controladores em um sistema hierárquico e tem como objetivo fornecer um controlador mestre na parte superior da rede e, assim, interconectar vários outros controladores. Esta abordagem é restrita ao controlador *Kandoo* e não é adequada para redes que utilizam outros controladores SDN.

DISCO [Phemius et al. 2013] propõe compartilhar o *status* da conexão de rede, mantendo a comunicação entre todos os controladores vizinhos. Além disso, esta abordagem requer que o código fonte das aplicações sejam atualizados no controlador *Floodlight*, o que torna a proposta completamente dependente deste controlador.

Apesar dos benefícios das propostas acima referidas, elas não se preocupam com a possibilidade de convergência com os vários tipos de controladores atualmente disponíveis, sendo restritos a um controlador específico, o que limita o potencial de heterogeneidade da rede.

Nessa perspectiva, na próxima seção examinaremos o sistema MISSIN, que é um inovador sistema de orquestração de infraestrutura SDN que abstrai a complexidade do gerenciamento de rede, permitindo: (*i*) a orquestração de novas infraestruturas SDN; (*ii*) visualização de eventos em tempo real; e (*iii*) interação entre o operador e dispositivos de rede. Além disso, MISSIn não depende de um controlador específico ou de ferramentas de terceiros, o que dá mais autonomia ao operador de rede.

3. MISSIn

A arquitetura do sistema MISSIn fornece ao operador de rede a autonomia para usar qualquer controlador para gerenciar a infraestrutura, executando a lógica de controle SDN dentro de uma entidade chamada *MISSIn Logic Unit*, que está no topo do controlador SDN. A Figura 1 fornece uma visão geral da proposta do MISSIn sob uma típica arquitetura SDN e aponta como a ferramenta interage com ela.

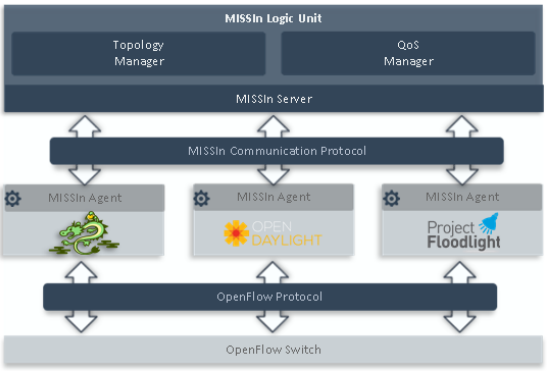


Figura 1. Visão geral da proposta MISSIn

Para ser capaz de realizar as operações propostas, o *Logic Unit* foi projetado com a capacidade de gerenciar vários controladores, além de permitir que estes coexistam. O uso desta arquitetura reduzirá a sobrecarga dos controladores SDN, induzida pelas aplicações, e também os pontos de falha.

A Figura 2 apresenta a arquitetura modular do MISSIn, enfatizando, em seus componentes principais, as funções de controle de rede, protocolos e interfaces.

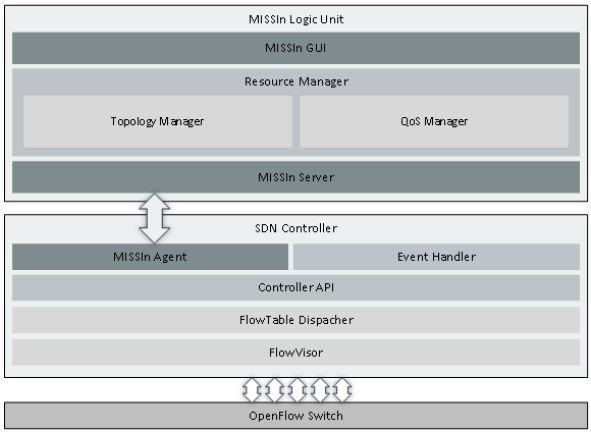


Figura 2. Arquitetura do sistema MISSIn

As subseções seguintes detalham os componentes principais da arquitetura do sistema MISSIn.

3.1. MISSIn Logic Unit

O *MISSIn Logic Unit* também implementa uma *GUI* (*Graphical User Interface*) onde o operador tem acesso a um mapa completo e intuitivo da infraestrutura. Isso facilita a comunicação com os dispositivos e também torna possível gerenciar as regras de fluxo de uma maneira que seja mais conveniente.

3.2. GUI

O *GUI* é um sistema web projetado para aplicar uma abstração na granularidade da infraestrutura SDN. A partir deste componente é possível interagir com os dispositivos da rede, gerenciando as aplicações providas pelo *Resource Manager*.

3.3. Resource Manager

O *Resource Manager* fornece, por padrão, um conjunto de mecanismos de controle de recursos (*Topology Manager* e *QoS Manager*), que implementa uma plataforma para suportar serviços inovadores através de recursos avançados que atualmente não estão disponíveis em controladores SDN tradicionais.

O *Topology Manager* é responsável pela obtenção de informações sobre a composição e o *status* da rede e, através de um diagrama da topologia, viabiliza a interação do operador com os dispositivos de modo que seja possível realizar tarefas de forma dinâmica e intuitiva. Desta forma, é possível gerir a rede de uma forma mais prática, utilizando as seguintes funções: (*i*) gerenciamento (consulta, inserção e remoção)

de fluxos e estatísticas da rede; e (*ii*) criação e manutenção de políticas de QoS fornecidas pelo *QoS Manager*.

O *QoS Manager* permite a criação de políticas de QoS com capacidades de reserva de recursos em caminhos de dados selecionados pelo operador de rede. Asssim, é capaz de fornecer QoS garantida, mantendo bons níveis de Qualidade de Experiência (*Quality of Experience* – QoE) para os usuários da rede.

3.4. MISSIn Server

O *MISSIn Server* é o componente responsável pela intermediação entre a GUI e o controlador SDN. Ele foi projeto para trabalhar de forma assíncrona, de modo que seja possível responder a várias requisições sem interferir na comunicação entre os demais controladores a ele conectados.

Para viabilizar a interoperabilidade entre os vários tipos de controladores SDN é necessário implementar uma camada de *middleware* entre o controlador e o *MISSIn Server*: esse componente é chamado de *MISSIn Agent*. Ele é responsável por implementar funcionalidades específicas no controlador SDN a partir de instruções enviadas pelas aplicações gerenciadas pelo *Resource Manager*. Para tal, se faz necessário uma implementação do *MISSIn Agent* para cada tipo de controlador SDN, sendo essa a única adaptação necessária para garantir a integração com a ferramenta, de modo que caberá ao MISSIn lidar com a complexidade do controlador alvo.

Uma introdução ao MISSIn foi previamente publicada em [Gomes et al. 2016], no entanto, a ferramenta está em contínuo desenvolvimento. Desta maneira, novos recursos serão disponibilizados à medida que são implementados e validados. Alguns dos recursos que atualmente estão em desenvolvimento são: (*i*) definição de semântica para modularização de componentes para permitir o desenvolvimento de novas aplicações para o *Resource Manager*; (ii) implementação de versões do *MISSIn Agent* para demais controladores SDN existentes (atualmente está disponível uma versão estável para o contrador *Ryu*[1] e uma versão experimental para o controlador *Floodlight*[2]); (*iii*) extensão do *QoS Manager* com o princípio de sobreprovisionamento de recursos [Logota et al. 2013]. O objetivo principal dessa estratégia é fornecer uma quantidade excessiva de recursos de rede (*i.e.* sobre-reserva de largura de banda através da classe de serviço) fornecendo informações com antecedência. Isso permite que o controlador gerencie sua operação de forma eficiente sem a necessidade de consultas e novos fluxos de sinalização.

4. Demonstração

Maiores detalhes sobre o MISSIn (código-fonte e manuais de instalação, além de um vídeo da ferramenta em funcionamento) podem ser encontrados no repositório oficial do projeto[3]. Uma máquina virtual pré-configurada também está disponível, facilitando o uso da ferramenta por novos usuários.

A demonstração consiste em apresentar algumas funcionalidades atualmente disponíveis no MISSIn para implementação de gerência dinâmica de fluxos e configuração

[1]https://osrg.github.io/ryu

[2]http://www.projectfloodlight.org/floodlight

[3]http://github.com/latarc/missin

de políticas de QoS e, para tal, serão considerados dois casos de uso. O primeiro diz respeito a configuração de fluxos para permitir uma simples comunicação entre dois dispositivos. No segundo caso, será considerado um cenário composto por vários dispositivos fazendo uso de serviço de *streaming* de vídeo. Para esta demonstração, serão utilizadas infraestruturas de rede SDN providas pelo emulador Mininet[4] e por *switches* TP-Link TL-WR1043ND (OpenWrt[5] v15.05 e OpenvSwitch[6] v2.3.90). Em ambos os casos, MISSIn orquestrará redes com múltiplos controladores.

No **caso 1**, será demonstrado como um operador de rede pode fazer uso da GUI para realizar operações básicas de gerência dos fluxos, tais como: criação de novos fluxos, consulta a fluxos existentes e remoção dos fluxos. Todas essas operações serão realizadas apenas por meio da interface gráfica. Será possível observar o comportamento do MISSIn em situações de falhas nos *links* de comunicação entre *switches*, onde fluxos relacionados a caminhos indisponíveis serão removidos e outros novos serão automaticamente adicionados através das funções da ferramenta.

No **caso 2**, considerar-se-á um típico cenário de esgotamento de recursos durante a solicitação de admissão de vários fluxos multimídia, o que impactará diretamente no QoE dos usuários. Através do recurso de gerência de QoS, será possível configurar, interativamente, políticas para fornecer garantias de QoS a determinados dispositivos.

5. Conclusão

Neste trabalho, propomos o sistema MISSIn, uma ferramenta de apoio para orquestração de infraestruturas SDN que trabalha de forma interativa e dinâmica. MISSIn é capaz de lidar com a abstração e complexidade do plano de controle das redes SDN, permitindo que novos recursos sejam integrados sem a necessidade de atualizações na infraestrutura de rede.

Até onde sabemos (e confirmado por estudos em trabalhos relacionados), não há evidências de propostas anteriores que forneçam suporte aos operadores de rede em lidar com a complexidade da heterogeneidade envolvida nas aplicações e requisitos de usuários, de forma dinâmica e interativa, conforme proposto por MISSIn. A próxima etapa deste trabalho será avaliar o desempenho do sistema MISSIn em um *testbed* SDN real.

6. Agradecimentos

Os autores gostariam de agradecer a PROPI/IFRN pelo apoio financeiro.

Referências

Choi, T., Lee, B., Kang, S., Song, S., Park, H., Yoon, S., and Yang, S. (2015). Iris-coman: Scalable and reliable control and management architecture for sdn-enabled large-scale networks. *J. Netw. Syst. Manage.*, 23(2):252–279.

E. Haleplidis, K. Pentikousis, S. D. J. H. S. D. M. and Koufopavlou, O. (2015). Software-Defined Networking (SDN): Layers and Architecture Terminology. RFC 7426, RFC Editor.

[4]http://http://mininet.org
[5]https://openwrt.org/
[6]http://openvswitch.org/

Gomes, C., Costa, K., Silva, J., and Silva, F. D. (2016). Missin: Um sistema interativo de gerência para infraestruturas sdn. In *XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2016). Anais do V Workshop de Pesquisa Experimental da Internet do Futuro (WPEIF 2016). Salvador, Brasil*, pages 21–24.

Hassas Yeganeh, S. and Ganjali, Y. (2012). Kandoo: A framework for efficient and scalable offloading of control applications. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 19–24, New York, NY, USA. ACM.

Huang, W.-Y., Chou, T.-Y., Hu, J.-W., and Liu, T.-L. (2014). Automatical end to end topology discovery and flow viewer on sdn. In *Proceedings of the 2014 28th International Conference on Advanced Information Networking and Applications Workshops*, WAINA '14, pages 910–915, Washington, DC, USA. IEEE Computer Society.

Logota, E., Campos, C., Sargento, S., and Neto, A. (2013). Advanced multicast class-based bandwidth over-provisioning. *Comput. Netw.*, 57(9):2075–2092.

Phemius, K., Bouet, M., and Leguay, J. (2013). Disco: Distributed multi-domain sdn controllers. *CoRR*, abs/1308.6138.

Salsano, S., Ventre, P. L., Lombardo, F., Siracusano, G., Gerola, M., Salvadori, E., Santuari, M., Campanella, M., and Prete, L. (2015). Mantoo - a set of management tools for controlling sdn experiments. In *Proceedings of the 2015 Fourth European Workshop on Software Defined Networks*, EWSDN '15, pages 123–124, Washington, DC, USA. IEEE Computer Society.

Schultz, J., Szczepanski, R., Haensge, K., Maruschke, M., Bayer, N., and Einsiedler, H. (2015). Opengufi: An extensible graphical user flow interface for an sdn-enabled wireless testbed. In *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*, pages 770–776.

Sezer, S., Scott-Hayward, S., Chouhan, P. K., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., and Rao, N. (2013). Are we ready for sdn? implementation challenges for software-defined networks. *IEEE Communications Magazine*, 51(7):36–43.

Shalimov, A., Nizovtsev, S., Morkovnik, D., and Smeliansky, R. L. (2015). The runos openflow controller. In *EWSDN*, pages 103–104. IEEE Computer Society.

Sharma, P., Banerjee, S., Tandel, S., Aguiar, R., Amorim, R., and Pinheiro, D. (2013). Enhancing network management frameworks with sdn-like control. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 688–691.

Voellmy, A., Wang, J., Yang, Y. R., Ford, B., and Hudak, P. (2013). Maple: Simplifying sdn programming using algorithmic policies. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 87–98, New York, NY, USA. ACM.

Salão de Ferramentas do SBRC 2017
Sessão Técnica 3
Visualização e Extração de Dados

DistViewer: Uma Ferramenta para Visualizar e Analisar o Código de Sistemas Distribuídos sob uma Nova Perspectiva

Fernando A. Teixeira[1], Hao-Chi Wong[2], José Marcos S. Nogueira[3], Leonardo B. Oliveira[3]

[1]UFSJ, Ouro Branco, Brasil

[2]Intel Corporation, Santa Clara, CA, EUA

[3]UFMG, Belo Horizonte, Brasil

teixeira@ufsj.edu.br, hao-chi.wong@intel.com, {jmarcos,leob}@dcc.ufmg.br

Abstract. *Despite major advances in recent decades in the field of compilers, developers and researchers still lack the tools to compile and analyze, in an integrated way, programs that communicate over the network. We propose a tool (DistViewer) able to provide users with an insight into the code of a distributed system, highlighting the interactions of the network via programs. The DistViewer is able to receive the source code of a distributed system, compile them and analyze them statically as if they were a single program. Our tool returns as output several graphs that represent the program flow control, data dependencies or relations between sending and receiving messages via network.*

Resumo. *Apesar dos grandes avanços nas últimas décadas na área de compiladores, os desenvolvedores e pesquisadores ainda carecem de ferramentas que permitam compilar e analisar, de forma integrada, programas que se comunicam via rede. Propomos uma ferramenta (DistViewer) capaz de fornecer aos usuários uma visão sobre o código de um sistema distribuído, destacando as interações dos programas via rede. O DistViewer é capaz de receber os códigos-fonte de um sistema distribuído, compilá-los e analisá-los estaticamente como se fossem um único programa. Nossa ferramenta retorna como saída diversos grafos que representam o fluxo de controle do programa, dependências de dados ou a relações entre envio e recebimento de mensagens via rede.*

1. Introdução

Novas tecnologias como a Internet das Coisas e Cloud Computing estão aumentando a importância dos sistemas distribuídos [Borgia 2014]. Estamos envolvidos por um grande números de sistemas embarcados em diferentes tipos de dispositivos e executando diferentes tipos de serviços. Há software embarcados em eletrodomésticos, automóveis e redes de sensores espalhados nas cidades, executando em hardware com diferentes capacidades e sujeitos a uma grande variedade de condições do ambiente. Programar esses sistemas se torna a cada dia mais desafiador, devido não somente ao seu grande volume, mas também sua diversidade.

Os compiladores têm um papel importante na análise de programas, detectando e prevenindo erros. Por exemplo, durante a fase de otimização de código, o compilador

pode usar técnicas de análise de código para alertar os programadores a respeito de defeitos no código ou vulnerabilidades de segurança tais como Buffer Overflow (BOF) e Integer Overflow (IOF) [Chess and West 2007]. Tais técnicas também podem ser usadas para corrigir defeitos funcionais ou de desempenho automaticamente (e.x., eliminação de código morto e melhoria de alocação de registradores). Através da detecção de defeitos no início do processo de desenvolvimento, quando é mais barato de corrigir, a análise estática de programas pode ajudar a prevenir problemas que seriam difíceis ou caros de detectar e corrigir durante os teste ou após a implantação do programa.

Apesar da análise estática ser fundamental para entender e aprimorar os sistemas distribuídos, a maioria das análises estáticas não foram projetadas para trabalhar nesse cenário. Então, elas analisam cada programa do sistema independentemente, e não tentam explorar sua natureza distribuída. Essa falta de visibilidade da interação entre programas força as ferramentas de análise estática a confiar em suposições que comprometem sua precisão. Essa falta de uma visão holística é lamentável, porque o conhecimento das interações entre programas poderia permitir que o compilador encontrasse defeitos devido à complexa interatividade de tais sistemas. Por exemplo, o compilador poderia avisar aos desenvolvedores sobre um SEND sem o correspondente RECV. Esse é um defeito fácil de corrigir durante a compilação. Entretanto, é difícil de ser detectado pelos testes ou durante a execução do sistema.

Para ilustrar este ponto, nós mostramos dois programas simples que enviam e recebem dados pela rede (Fig. 1). Neste exemplo, se o valor 7 for passado para o primeiro `scanf`, o último SEND no programa P_A (Fig. 1a – linha 13) não terá nenhum RECV correspondente no programa P_B. Se o compilador olhar para cada um desses programas separadamente, ele não detectará nenhum problema. Entretanto, se o compilador analisasse esses dois programas como um única entidade, ele poderia tentar casar cada SEND e RECV e, então, prover alarmes onde esse casamento não por possível.

```
1  int sd, code, size;
2  char msg[MAX];
3  sd = getMyServerSocket();
4  scanf("%d",&code);
5  if(code == 7){
6    strcpy(&msg,"78begin");
7    send(sd, &msg, 7, 0);
8  }
9  scanf("%s",&msg);
10 s = strlen(msg};
11 send(sd, &msg, s, 0);
12 scanf("%s",&msg);
13 send(sd, &msg, s, 0);
```

(a) Programa A - P_A

```
1  int sd;
2  char msg[MAX];
3  sd = getMySocket();
4  recv(sd,&msg,MAX,0);
5  if(msg[0]==7){
6    printf("Begin");
7    recv(sd,&msg,MAX,0);
8  }
9  else {
10   printf("%c\n",msg[1]);
11   recv(sd,&msg,MAX, 0);
12 }
```

(b) Programa B - P_B

Figure 1. Defeito de Interação via Rede

Em [Teixeira et al. 2015] nós mostramos que é possível ligar SENDs e RECVs correspondentes presentes em diferentes programas de um sistema distribuído durante o tempo de compilação, e prover aos desenvolvedores informações úteis a respeito da comunicação inter-programa, permitindo, assim, que ferramentas de análise estática possam utilizar

uma visão integradas dos sistemas distribuídos. O sistema proposto em [Teixeira et al. 2015] exige um conhecimento avançado de compiladores para ser usado ou estendido.

Nesse trabalho, nós projetamos, implementamos e testamos uma ferramenta que chamamos de *Distributed System Code Viewer (DistViewer)*[1] – *Distributed System Code Viewer* que encapsula e estende a solução proposta em [Teixeira et al. 2015], de forma que possa ser usada diretamente por desenvolvedores e projetistas de protocolos de rede, via web, sem a necessidade de conhecimentos avançados de compiladores ou de configuração de infraestrutura de compilação específica. O DistViewer permite receber dois programas em C que se comunicam via rede. A ferramenta compila os códigos fontes recebidos utilizando o compilador *clang* do LLVM [Lattner and Adve 2004], analisa o código intermediário, e exibe como resultado grafos que destacam a interação via rede dos programas. Tudo isso é feito de maneira automática e intuitiva através de uma interface *web*. Dessa maneira, o usuário pode analisar seu código sem a necessidade de instalar ou configurar compiladores e outras bibliotecas em sua máquina. O usuário da ferramenta pode visualizar os grafos na interface *web* ou baixá-los em formato *pdf*. Além disso, o DistViewer gera uma visão resumida do sistema – Network Programing Slice (NetSlice) – que corresponde a 5% do programa original.

O restante desse artigo está organizado das seguinte maneira. A seção 2 descreve conceitos de linguagem de programação usados nesse artigo. A seção 3 descreve as funcionalidades e a arquitetura do DistViewer. A seção 4 informa como será feita a demonstração da ferramenta e a URL de acesso. Nós concluímos o artigo na seção 5.

2. Conceitos Fundamentais

Há diversas soluções para analisar o código de sistemas, e a maioria se baseiam na Análise Estática [Chess and West 2007], na Análise Dinâmica [Serebryany et al. 2012], ou na combinação de ambas. Na Análise Estática, o sistema é inspecionado antes do programa ser implantado. A vantagem é que não há sobrecarga durante a execução do sistema. Por outro lado, a análise não possui informações que só estarão disponíveis em tempo de execução. Na Análise Dinâmica, por outro lado, o sistema é, sim, executado. Agora, a técnica pode tirar proveito das informações só disponíveis em tempo de execução. Isso, por um lado, mitiga a questão dos falso-positivos, comuns à Análise Estática. Mas, seus resultados são pertinentes apenas às entradas testadas e, assim, não se pode tirar conclusões acerca do comportamento geral do programa. Em razão de suas naturezas complementares, é comum o emprego da Análise Híbrida, ou seja, a combinação de ambas.

Diversos ferramentais são utilizados durante a análise de sistemas. Dentre eles, dois são especialmente importantes no contexto deste trabalho: o Grafo de Fluxo de Controle (CFG) e o Grafo de Dependência (DG) [Chess and West 2007]. O CFG é um grafo dirigido que representa as possíveis sequências de instruções que são processadas durante a execução de um programa. Um vértice do CFG é chamado um bloco básico. Um bloco básico é um conjunto maximal de instruções que sempre são executadas em sequência. Existe uma aresta entre um bloco básico b_1 e um bloco básico b_2 se, e somente se, o programa pode fluir de b_1 para b_2. O DG é uma estrutura de dados usada pelo compilador para modelar as dependências entre os dados e as instruções de um programa. Um DG possui um vértice para cada variável e cada operação em um programa. Em um DG há

[1]Disponível em `http://cuda.dcc.ufmg.br/dsa/`

uma aresta entre cada variável u e uma instrução i se i representa uma instrução que usa u. Na sua forma padrão, CFGs e DGs não são capazes de modelar o fluxo de controle e a dependência de dados entre múltiplos programas. A seguir, descrevemos uma solução que endereça esse problema.

2.1. Inferência de Canais de Comunicação

Inferência de Canais de Comunicação é o problema de determinar canais de comunicação entre programas distribuídos. No nosso contexto, o problema de inferência de canais de comunicação recebe duas entradas: o código de dois programas, P_1 e P_2, que se comunicam. A solução desse problema consiste em um conjunto de pares C, que relaciona vértices especiais presentes em P_1 e P_2. Esses vértices especiais são chamados de SEND (emissores) e RECV (receptores). Vértices da primeira categoria emitem mensagens, enquanto vértices da segunda categoria consomem essas mensagens. Se C contém um (s, r), então nós sabemos que o emissor s *pode* enviar uma mensagem que o receptor r consome. Entretanto, se tal par não está presente em C, então nós sabemos, com certeza, que s *não pode* enviar uma mensagem diretamente para r. Fig. 2 mostra uma solução para o problema de inferência de canais de comunicação. Cada linha tracejada é um possível canal de comunicação, como inferido pela análise. Para analisar um sistema distribuído como um todo, nós precisamos trabalhar com CFG e DG que transcendem as fronteiras dos programas. Para tanto, podemos utilizar a Inferência de Canais de Comunicação para criar Distributed Control Flow Graphs (DCFGs) e Distributed Dependence Graphs (DDGs) conforme definidos a seguir.

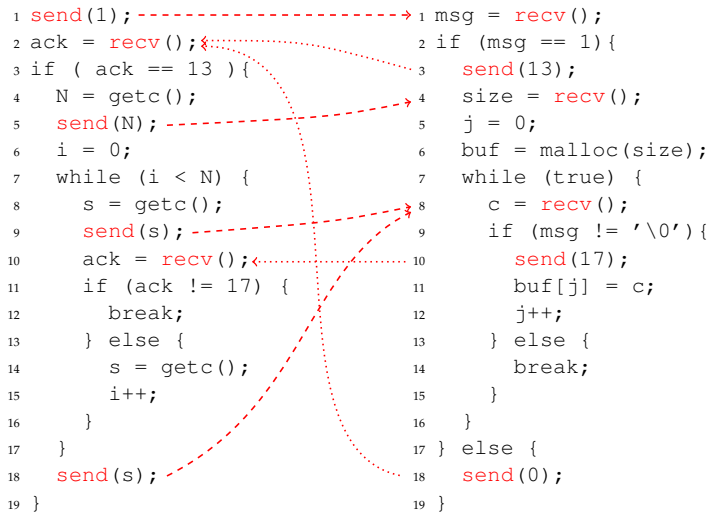


Figure 2. Inferência de Canais de Comunicação por [Teixeira et al. 2015]

2.2. Distributed Control Flow Graph (DCFG)

Seja $\{\mathcal{C}_1, \mathcal{C}_2\}$ um par de CFGs que contituem um sistema e D o DCFG resultante. D contém C_1 e C_2 como um subgrafo. Arestas inter-programa que conectam C_1 e C_2 são então adicionadas ao D: para cada par de vértices SEND e RECV que podem se comunicar, nós adicionamos uma aresta do primeior para o último. Isto é, para cada par de vértices $s_i \in \mathcal{C}_1$ e $r_j \in \mathcal{C}_2$, se há uma sequência de execução na qual uma mensagem enviada por s_i

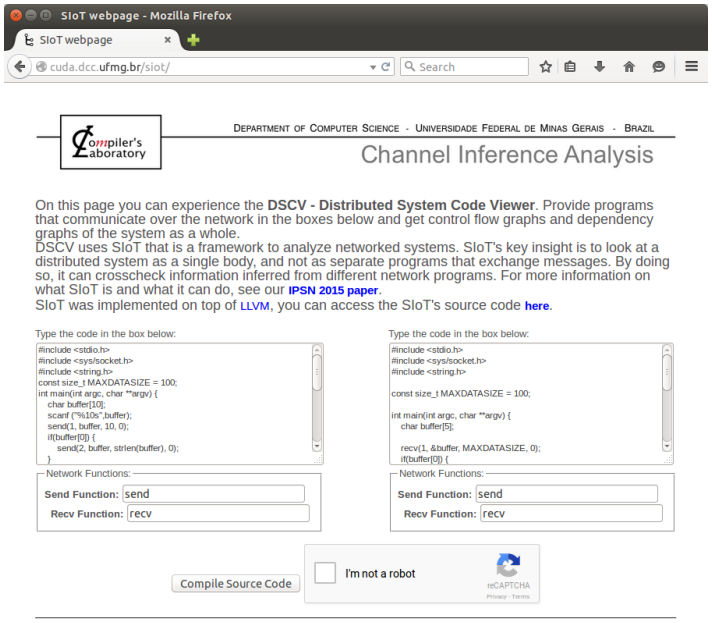


Figure 3. DistViewer Tela Inicial.

alcança r_j, nós adicionamos ao D uma aresta inter-programa de s_i para r_j. Para cada par de vértices $s_k \in \mathcal{C}_2$ e $r_t \in \mathcal{C}_1$, se há uma sequência de execução na qual uma mensagem enviada por s_k alcança r_t, adiciona-se a D uma aresta inter-programa de s_k to r_t.

2.3. Distributed Dependence Graph (DDG)

A partir do DCFG, podemos criar um grafo de dependências distribuído. Tal grafo pode ser construído a partir da união dos grafos de dependências individuais de cada módulo, via uma estratégia semelhante àquela que usamos para unir CFGs. Cada programa do sistema possui seu próprio grafo de dependência. Para cada instrução de acesso a rede, é criado um vértice no grafo para representar essa operação. O vértice é marcado como um nó de rede de leitura ou de escrita e inserido em uma lista correspondente. Insere-se, então, uma aresta de dependência entre o vértice de rede encontrado e os vértices correspondentes no outro programa de acordo com o DCFG do sistema construído anteriormente. Dessa forma, o grafo de dependências final é a união dos grafos de cada programa. Essa estrutura representa o grafo de dependências do sistema distribuído como se tal sistema fosse um único programa.

3. DistViewer - Funcionalidade e Arquitetura

O DistViewer pode ser usado para auxiliar desenvolvedores de sistemas distribuídos e/ou projetista de protocolos de rede. A partir do grafos de sistemas distribuídos gerados, o projetista ou o desenvolvedor podem, por exemplo, (i) verificar erros semânticos, auxiliados pelos grafos que sumarizam e destacam a interação via rede; (ii) realizar otimizações no código tendo como base a interação entre elementos de rede; e (iii) verificar caminhos entre entradas de usuários e vértices de memórias para achar dependências de dados.

A atual versão do DistViewer recebe como entrada o código fonte de dois programas em C que comunicam via rede (Fig. 3). A ferramenta, então, compila os códigos

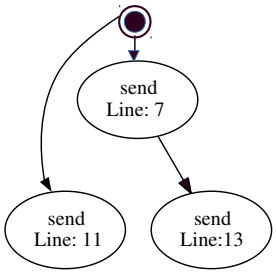


Figure 4. Send-Graph de P_A.

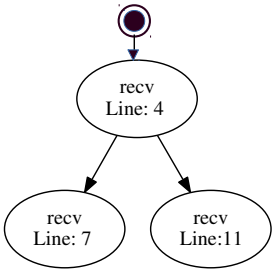


Figure 5. Receive-Graph de P_B.

fonte usando o clang do LLVM, infere os canais de ligação entre os programas e faz a junção dos *bytecodes* gerados. A partir dessa visão global dos programas, o DistViewer gera vários grafos que modelam o comportamento dos programas em relação ao acesso à rede e à memória (Send-Graph, Receive-Graph, *Distributed Dependency Graph* e o NetSlice). Ao final, o DistViewer formata e exibe os grafos gerados via interface *Web*.

Inicialmente o DistViewer gera como saída uma visão do grafo de fluxo de controle de SENDs (Send-Graph) e RECVs (Receive-Graph) de cada programa. Por exemplo, a Fig. 4 mostra o Send-Graph do programa P_A (Fig. 1a) e o Receive-Graph (Fig. 5) do programa P_B (Fig. 1b). Com grafos desse tipo, desenvolvedores ou projetistas de protocolos passam a ter uma visão geral do fluxo de comunicação via rede, sem detalhes de implementação. Tal abstração torna a análise do protocolo mais simples e auxilia na detecção de erros e na otimização dos protocolos de rede.

Em seguida, o DistViewer gera o *Distributed Dependency Graph* e, a partir dele, é construído o NetSlice que é uma fatia de programa (*programming slice*) que destaca a interação entre memória e rede inter-programas. Essa fatia de programa é representada por um grafo que destaca os vértices que representam as operações de rede (i.e., SENDs e RECVs) ou os endereços de memória que contribuem para o fluxo de informação através da rede. A Fig. 6 mostra o NetSlice dos programas P_A e P_B (Fig. 1). Mais formalmente, dado um DDG G, nós definimos um NetSlice N associado como segue: Para cada vértice $v \in G$, nós adicionamos um vértice v ao N se e somente se: (i) v é um SEND, ou (ii) v é um RECV, ou (iii) v é um nó de *memória*. Arestas em N correspondem a caminhos no G original. Para cada par de vértices $u, v \in G$, nós adicionamos uma aresta $\overrightarrow{u'v'}$ ao N se, e somente se: (i) há um caminho p de u para v em G, e (ii) p não contém nenhum outro vértice em G. Note que a visão do sistema como um todo permite que o usuário visualize qual SEND de um programa pode trocar informações com qual RECV de sua contraparte. Podemos dessa forma, por exemplo, verificar que há SENDs sem um RECV correspondente ou vice-versa, como acontece em nosso exemplo. O "SEND Line:13" (Fig. 6) não possui um RECV corresponde, o que nos permite detectar, a partir do grafo, o *bug* do nosso exemplo. Adicionalmente, essa visão permite que sejam verificados as dependências entre diferentes regiões de memória, mesmo se elas estão localizados em nós de rede independentes, como acontece entre a Memory 1 e a Memory 4.

3.1. Arquitetura

A arquitetura do DistViewerpode ser divida em 5 módulos (Fig. 7): (i) entrada de dados; (ii) compilação e geração dos *bytecodes*; (iii) inferência de links entre programas; (iv) geração dos dados dos grafos; e (v) formatação (em *pdf* e *html*) e visualização dos grafos.

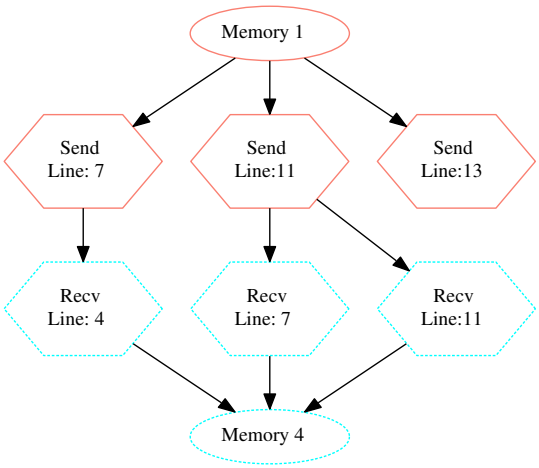


Figure 6. NetSlice de P_A e P_B. Vértices de P_A são representados com linhas contínuas (salmão) e vértices de P_B com linhas tracejadas (ciano). Através desse grafo fica fácil perceber que o SEND da linha 13 não possui um RECV correspondente.

A *Entrada de Dados* processa os códigos fonte e os dados de configuração e realiza as validações necessárias. O módulo de – *Compilação* – compila os códigos fonte repassados como entrada e gera os *bytecodes* LLVM correspondentes. Em seguida, o módulo de *Inferência de Comunicações* gera um *bytecode* único do sistema através da inferência de ligações entre SENDs de um programa com RECVs de outro programa. Usando como entrada *bytecode* único, o módulo de *Geração de Grafos* gera os grafos Send-Graph, Receive-Graph, Grafo de Dependência Distribuído, além da visão resumida das interações via rede que chamamos aqui de NetSlice. Finalmente, o módulo de *Formatação e Visualização* processa os grafos gerados exibi-os via *Web* em formato *pdf* ou *html*.

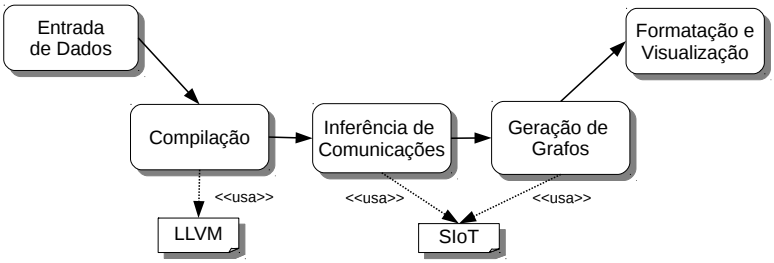


Figure 7. Arquitetura.

3.2. Testes

Para avaliar a capacidade do DistViewer de processar programas reais nós conduzimos testes em seis aplicações do *ContikiOS* [Dunkels et al. 2004]. A Tabela 1 sumariza nossos resultados. Além de demonstrar a viabilidade de executar o DistViewer usando programas reais, pode-se observar que o NetSlice reduziu o número de vértices do DDG para 5%. Essa redução é possível porque a fatia de programa gerada foca nas operações de acesso à rede (SENDs e RECVs) e de acesso à memória, que são os principais elementos a serem analisados neste tipo de sistema.

Table 1. DDG Vertices: Baseline versus Grafo NetSlice do DistViewer

Programa	Baseline	DistViewer	%
netdb client/server	95,656	5,070	5.30%
ping / new-ipv6	75,899	3,719	4.90%
ipv6-rpl-collect udp-sender/sink	78,365	3,918	5.00%
ipv6-rpl-udp client/server	77,128	3,934	5.10%
udp-ipv6 client/server	78,365	3,918	5.00%
coap-client / rest-server	82,647	4,711	5.70%

4. Demonstração e URL

Para demonstrar o DistViewer usaremos códigos de exemplo e instigaremos os participantes a entrar com códigos a serem testados. O código fonte, os manuais e a documentação da ferramenta podem ser obtidos na URL `http://cuda.dcc.ufmg.br/dsa/`. Link do Vídeo: `http://youtu.be/G0EGKfZKrl4`

5. Conclusão

Nesse trabalho, apresentamos o DistViewer: uma ferramenta que fornece aos usuários uma visão inter-programa sobre o código de sistemas distribuídos destacando as interações dos programas via rede. O DistViewer é capaz de receber códigos fontes em C de programas de um sistema distribuído, compilá-los, analisá-los suas interações via rede e retornar como saída diversos grafos como: grafos de envio de mensagem, grafos de recebimento de mensagens, grafo de dependência de dados entre programas dos sistema distribuição e o NetSlice que resume a interação entre memória e operações de rede. Entre potenciais usuários, acreditamos que o DistViewer será útil tanto para desenvolvedores de sistemas distribuídos como para projetistas de protocolos de rede.

Agradecimentos. Agradecemos ao financiamento parcial do CNPq e Fapemig.

References

Borgia, E. (2014). The internet of things vision: Key features, applications and open issues. *Computer Communications*, 54:1–31.

Chess, B. and West, J. (2007). *Secure Programming with Static Analysis*. Addison-Wesley Professional, first edition.

Dunkels, A., Gronvall, B., and Voigt, T. (2004). Contiki - a lightweight and flexible operating system for tiny networked sensors. In *International Conference on Local Computer Networks (LCN)*. IEEE.

Lattner, C. and Adve, V. S. (2004). LLVM: A compilation framework for lifelong program analysis & transformation. In *International Symposium on Code Generation and Optimization (CGO)*. IEEE.

Serebryany, K., Bruening, D., Potapenko, A., and Vyukov, D. (2012). AddressSanitizer: a fast address sanity checker. In *Annual Technical Conference (ATA)*. USENIX.

Teixeira, F. A., Machado, G. V., Pereira, F. M., Wong, H. C., Nogueira, J., and Oliveira, L. B. (2015). SIoT: Securing the internet of things through distributed system analysis. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, pages 310–321. ACM.

SLkit: An R package for property extraction and analysis of multiple Sensing Layers*

Fabrício Ferreira[1], Thiago H. Silva[2], Antônio A. F. Loureiro[1]

[1]Department of Computer Science
Federal University of Minas Gerais
Belo Horizonte, Brazil

[2]Department of Informatics
Federal University of Technology - Paraná
Curitiba, Brazil

{fabricio.silva,loureiro}@dcc.ufmg.br, thiagoh@utfpr.edu.br

Abstract. *The popularisation of smartphones and the increased use of mobile applications allow users to become not only consumers but also data producers. Data shared voluntarily by users through mobile applications open unprecedented opportunities to generate useful knowledge, identify and solve issues, and provide new services. Due to the complexity of urban phenomena and the large volume of data available, stages such as modelling, preprocessing and analysis of sensing layers can be time-consuming. To help to tackle that issue, this study presents an R package intended to give support to researchers regarding decision making and evaluation of sensing layers. It provides functions for property extraction and multilayered analysis which can be customised according to one's project needs, helping to leverage new applications that explore the concept of sensing layers.*

1. Introduction

The number of people who live in urban areas outnumbers, since 2009, the portion of rural residents. This difference continues to increase, mainly in developing countries where people move to the city in search for a job opportunity and better quality of life [DESA 2015]. As a result government authorities have focused on projects that face challenges associated with urban sprawl regarding basic services such as energy, housing, transportation, waste management, water and sanitation [Khatoun and Zeadally 2016].

Another fact to be considered is the evolution of the use of mobile gadgets and how it is changing the way people relate to the digital and physical world. Smartphones are equipped with powerful sensors used in various applications. Some of these use geographic location service to improve user experience, allowing users to generate a large volume of data. This data can contain information capable of characterising personal preferences (e.g., places one enjoys going) as well as collective behaviour within a specific location (e.g., how people move around a neighbourhood). In this scenario, users are considered sensors themselves, composing a Participatory Sensor Network (PSN),

*The code and documentation are available on https://github.com/FdeFabricio/POC. The project is licensed under the GPL. Any contribution/comment is welcome.

which have a wider coverage and lower cost than traditional Wireless Sensor Networks (WSN) [Silva et al. 2014a]. Hence urban solutions which take into account such data have been shown to be more effective and more democratic.

The large volume of data regarding diverse aspects of a city represents an unprecedented opportunity for assimilating knowledge, identifying and solving problems, besides providing new services. In order to achieve that, sophisticated methods of data analysis are required, capable of dealing with the heterogeneity of data sources, a common challenge of this context. A possible approach, discussed in Section 2, is by modelling each dataset as a Sensing Layer (SL), which *"consists of data describing specific aspects of a geographical location"* [Silva et al. 2014b]. The majority of urban phenomena have substantial complexity and could not be described using a single layer. Urban mobility, for instance, can be influenced by the weather, topography, big events, distribution and level of public transportation service, income distribution, among others. Therefore, it is fundamental to adopt a multilayered approach in order to characterise complex phenomena and maybe try to predict them more accurately. However, the use of multiple layers face challenges pointed out by [Silva et al. 2015] such as ensuring spatiotemporal consistency between all related layers, compiling data by location and/or users and comparing data with distinct lifespan.

Pre-processing and analysis of sensing layers are important steps since both potentially valuable insights and prevent inappropriate use of sensing layers, such as spatiotemporal incompatibility and illusory correlations. This stage is often done manually or through specific scripts, which are not commonly reused seamlessly in other projects, becoming a repetitive and redundant effort. Thus, this work presents a tool capable of automating important processes of spatial and temporal properties extraction and analyses on multiple SLs. The tool consists of an R package and it is intended for researchers who wish to save time on projects involving a considerable number of spatiotemporal datasets.

This paper is organised as follows. Section 2 defines Sensing Layers, how they can be used and their challenges. Section 3 presents the related work. Section 4 discusses temporal and spatial property extractions on the package. Section 5 discusses multilayered analyses. Finally, Section 6 points out the conclusions and future work.

2. Sensing Layers and its challenges

A SL is a set of data related to the same domain, describing certain aspects of a particular geographic region [Silva et al. 2014b]. For instance, layers may contain information about weather, traffic condition, social networks, check-ins, etc. Such data can be obtained from a variety of sources, such as PSNs, Web services, or even conventional WSNs. In general, they are composed of parameters such as: timestamp in which data were measured or collected; geographical location it is inserted into; agent identification (e.g., sensors or user ID of a social network); elements describing domain specificities and the value of the measured variable.

The use of SLs allows continuous monitoring of an area over time on assorted aspects, as well as information retrieval that would not be possible to conduct by analysing each dataset separately. Such representation makes it easier to investigate his-

torical data and identify patterns. The application of individual sensing layers already shows value, as mentioned in Section 3. However, the use of multiple layers allows data contextualisation which consequently allows a better understanding of complex phenomena.

There are challenges in using multiple SLs though. The lack of spatial or temporal correspondence may lead to an illusory correlation between different layers. In order to conduct correct analysis it is fundamental that the datasets are in a compatible format and that there is spatiotemporal correspondence, i.e., the data ought to be from the same area and period of time. It is also possible that the data distribution is not uniform lacking data in specific time frames (late at night or bank holidays, for example) or in specific areas (e.g., industrial neighbourhood and suburbia), which makes it problematic to draw conclusions or to validate hypotheses. Typically spatial and/or temporal resolution, i.e., how frequent a data is updated and how many square meters it represent, are not equivalent which also can impact on the multilayered correlation. Depending on the context, different conditions such as the above mentioned must be satisfied to legitimise a multilayered study, which proves the importance of the property extraction and analysis stage.

3. Related work

It is possible to extract information when conducting data analysis on a single layer. However, the lack of complementary data can restrict the study case or even lead to significant error. For example in 2008 Google launched Google Flu Trends (GFT), a Web service to estimate the number of flu cases by using data from its search engine. The initial estimates provided overshot more than 50 % of the number of cases and even after adjustments the error was still greater than 30 % [Lazer et al. 2014]. On the other hand, the GFT worked highly accurate when used in conjunction with data from the Centers for Disease Control and Prevention (CDC).

[Silva et al. 2014b] present a study about using PSNs as SLs. Using two-layered examples, the authors discuss the process of data collection, modelling and operations for information retrieval. The first case uses data from Instagram and Foursquare to identify points of interest in the city of Belo Horizonte, Brazil. The second correlates sentiments found in comments on Foursquare about establishments in a particular area with the average income of its inhabitants, indicating the absence of negative feelings in areas with greater purchasing power.

[Bakhshi et al. 2014] analysed Yelp reviews, correlating them with weather, demographics and local characteristics. In this study, the authors highlighted factors which directly influence clients emotions, which also indirectly affect their evaluation.

[Machado et al. 2015] use multiple Sensing Layers to analyse urban mobility in six cities (New York, Chicago, Los Angeles, Paris, London and São Paulo). Based on weather data from the Weather Underground service in addition to a database of check-ins made in these cities, the authors attempt to find a relationship between climate and urban behaviour analysing approximately 120 days. The findings show that there is a phase transition phenomenon, supported by the correlation between temperature variation and mobility pattern in Paris and London. In other cities where such phenomenon

was not well structured, the variation of the movement pattern was still identified when considered small regions within the city.

In the scope of urban mobility, there are studies that analyse databases on traffic flow and its correlation with social networks. [Tostes et al. 2013] use traffic data from Bing Maps to predict traffic conditions, such as congestion, with data from the preceding week. [Ribeiro et al. 2014] try to understand if Foursquare and Instagram data can be used to understand traffic in cities. This work shows that there is a great correlation between behavioural data of social networks (social sensor) and traffic conditions, being really useful, again, for predictions.

Although there are R packages created for data science projects intended to pre-process geospatial data and even make map plots, there is none for sensing layer analysis. The package presented in this work relies on generic R packages such as dplyr, ggplot2, rgdal, rgeos and ggmap and provides functions to be used and customised, if needed, on multilayered data projects.

4. Property Extraction

Different types of data can be collected from distinct sources. Temperature, for example, can be obtained by distributed sensors in a city, meteorological stations, prediction by satellite image analysis, PSNs, among others. Using all these sources to represent the same variable might configure an unnecessary effort, which could impact directly on the study complexity. Therefore, it would be necessary to extract characteristics from each dataset and select the one, or those, that best meet the purpose of the work. Another important aspect is to evaluate the spatial and temporal coverage of the data and its degree of repeatability: you may have a large amount of data, but 20% of those may be sufficient to represent that variable reliably.

For the reasons given above, it proves necessary to quantify characteristics of SL to serve as a basis for decision-making, in addition to performing analyses, discussed in Section 5.

The term property, in this work, is any spatial or temporal characteristic of a Sensing Layer that can be measured and compared among different layers. It was not found in the literature a list of properties that can be extracted from a SL. Therefore this work presents a set of properties, divided into temporal and spatial, selected based on the methodology of the related work. Not that this list can also be expanded when necessary.

4.1. Temporal Properties

This set of properties is a mechanism to characterise how data behave considering time as a factor. This work implements the functions presented below. The documentation, parameter listing, output plots and examples of use are in the project repository[1].

- **Temporal Coverage (tpCoverage):** temporal interval the data is inserted into. This function receives a dataframe, sweeps all its data and returns a vector with the earliest and latest timestamp.

[1] https://github.com/FdeFabricio/POC

- **Temporal Distribution (tpDistribution):** how much the data is distributed through time. This function divides the temporal coverage according to an input resolution and returns the percentage of it which has data associated with. For example for a crime dataset, if the time resolution is set as ”daily”, the function divides the data into separate days and return the number of days that have at least one crime record over the total of days.
- **Refresh Rate (refreshRate):** how often the data is updated. This function returns the average of the time difference between consecutive measurements.
- **Temporal Popularity (tpPopularity):** how much data are in each time unit considering a given resolution as input (hour of the day, day of the week, etc.) The function returns a histogram with the percentage of total data of each time interval.

4.2. Spatial Properties

These properties quantify the relationship with the geographic space. This work implements the functions presented below. The documentation, parameter listing, output plots and examples of use are in the project repository[2].

- **Spatial Coverage:** area the data is inserted into. This function analyses the latitude and longitude values of an input dataframe and returns the extreme coordinates forming a bounding box. It can also return a plot with the data and the bounding box on a map.
- **Spatial Distribution:** how much the data is distributed through space. This function divides the spatial coverage into a number of rectangles (given as input) and returns the percentage of them that has data associated with. For example for the crime dataset with a spatial coverage of 100 km^2, if the spatial resolution is 50, this function divides the space into 2 km^2 rectangles and returns the number of rectangles that have at least one crime record inside it over the total of rectangles.
- **Spatial Popularity:** this function returns a heat map, highlighting areas that have more data (high popularity) and empty ones (low popularity).

5. Multilayered Analysis

After analysing each dataset individually it is important to see how the selected layers relate to each other. A multilayered analysis intends to attest if the variable represented by one layer have any effect on another layer. Taking as an example two datasets of bicycle rental and precipitation level, one can try to verify if the volume of rain influences the use of such transport mode. To do so it is necessary to calculate the correlation of the two variables (rainfall in mm and number of rents, for example), which can be negative (the more it rains, the fewer people rent bikes), positive (the more it rains, the more people rent bikes) or no correlation (the rain does not affect the number of bicycle rentals).

To run a multilayered correlation, one must make sure that there is no incongruity between layers. By adopting data on the number of rented bicycles during the month of January and the rainfall index of August, one makes the mistake of trying to

[2]https://github.com/FdeFabricio/POC

```
$temporal
             checkin  instagram weatherUn noiseTube
checkin   1.00000000 0.99992646         1         0
instagram 0.99999316 1.00000000         1         0
weatherUn 0.03718432 0.03718184         1         0
noiseTube 0.00000000 0.00000000         0         1

$spatial
            checkin instagram noiseTube
checkin   1.0000000 0.9965368         0
instagram 0.9989773 1.0000000         0
noiseTube 0.0000000 0.0000000         1
```

Figure 1. Example of a four-layered STIA output

relate data that is time-separated and therefore could not have any causal relationship. The same mistake would occur if the layers correlated were separated in space (e.g., London bicycle rental data and Singapore's rainfall index).

To ensure that scenarios similar to those mentioned above do not occur, it is necessary to extract temporal and spatial properties of sensing layers and compare them. The package provides a SpatioTemporal Intersection Analysis (STIA), which aims to measure the level of intersection between different layers regarding spatial and temporal data. The function receives the different layers as parameters and returns a intersection matrix with the percentage of intersection between each layer. The result can be used to trim a layer when there is any incompatibility or even for decision making (to test if a certain dataset is suitable for the actual study).

Figure 1 presents an output of a STIA of four layers: Foursquare check-in data, georeferenced Instagram posts, precipitation by Weather Underground and noise level by Noise Tube. The resulting matrix shows that: 1) weatherUn has no geospatial data; 2) checkin and instagram layers are from the same area and period of time (spatiotemporal intersected); 3) the time period of checkin and instagram layers represents 3.7% of weatherUn's complete time interval; 4) noiseTube layer has no spatial nor temporal intersection with any other layer. Therefore, for a study with such layers, one would use only part of the weatherUn layer and none of the noiseTube.

Other analyses can be conducted considering the properties extracted. For instance, by extracting the seasonality and the temporal coverage of a layer, it's possible to identify if there are redundant data. Temporal and spatial popularity can be used to understand how users behave in a given social network. [Silva et al. 2013] identified, for instance, that Instagram data presented typical peaks at mealtimes, which is a valuable information when considering associating this data with other sources. Spatial popularity can be useful in a congestion study since PSNs such as Waze do not have

much data in the periphery nor small-sized cities. This means that any prediction will take into account only evidence from the city centre and main streets, which can be problematic.

Although the quality of a data source is used to decided if a layer will be considered in a project or not, it is generally difficult to determine an approach to measure it since it depends on the idiosyncrasy and the aims of a particular study. For example, a researcher can make a decision based on refresh rate when the variable measured constantly changes (e.g., bus position). Or wishing to create a robust predictive traffic model, one would choose layers with higher spatial and temporal distribution which, in other words, have data disperse on more parts of a the city (e.g., city centre and residential areas) and in different situations (e.g., peak time, night time, weekends, holidays). It is also possible that one would choose a layer with fewer data to minimise computational complexity. For example, in a study where no more than the average daily temperature is required, there is no need for a layer with temperature measurements every 5 minutes.

Different analyses can be conducted for each research and it depends entirely on the scope, selected layers and study goals. This package was thought to be incremented as necessary and free to any contribution. Since it is hosted on GitHub, one can download this package, import it to its project, make alterations to adapt the functions to the context of their study and even share such modifications back with the community. In addition, this package not only helps on the process of sensing layer research but also promotes collaboration between academics in an open source platform.

6. Final Remarks

This paper described a package developed in R to give support on the stages of modelling, pre-processing and analysis of sensing layers. It presents a set of functions capable of performing spatial and temporal property extraction and multilayered analysis intended to minimise effort on such studies dealing with sensing layers. The documentation of the project presents an installation tutorial, description of the functions and examples of their use. The package may need adjustments to adapt to the context and objectives of a given project. For this reason, an open source package becomes suitable since different researchers can make changes to the code and contribute to its improvement.

For future work it would be interesting to evaluate the level of acceptance of this package and if users are benefiting from it. Licensing the code with the GNU General Public License and providing the project on a collaborative development platform such as GitHub is a good alternative to engage collaborators and receive feedback.

Another possibility of future work includes the study of scenarios other than urban phenomena in order to attest the necessity of other property extractions. The package may need adjustments to deal with layers that have no geographical or temporal data since currently those are eliminated from the methods implemented.

References

Bakhshi, S., Kanuparthy, P., and Gilbert, E. (2014). Demographics, weather and online reviews: A study of restaurant recommendations. In *Proceedings of the 23rd Inter-*

national Conference on World Wide Web, WWW '14, pages 443–454, New York. ACM.

DESA (2015). *World Urbanization Prospects: The 2014 Revision*. United Nations, Department of Economic and Social Affairs, Population Division, New York.

Khatoun, R. and Zeadally, S. (2016). Smart cities: Concepts, architectures, research opportunities. 59(8):46–57.

Lazer, D., Kennedy, R., King, G., and Vespignani, A. (2014). The parable of google flu: Traps in big data analysis. *Science*, 343(14 March):1203–1205.

Machado, K., Silva, T. H., de Melo, P. O. V., Cerqueira, E., and Loureiro, A. A. (2015). Urban mobility sensing analysis through a layered sensing approach. In *2015 IEEE International Conference on Mobile Services*, pages 306–312. IEEE.

Ribeiro, A. I. J. a. T., Silva, T. H., Duarte-Figueiredo, F., and Loureiro, A. A. (2014). Studying traffic conditions by analyzing foursquare and instagram data. In *Proceedings of the 11th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, pages 17–24, New York. ACM.

Silva, T., Vaz De Melo, P., Almeida, J., and Loureiro, A. (2014a). Large-scale study of city dynamics and urban social behavior using participatory sensing. *Wireless Communications, IEEE*, 21(1):42–51.

Silva, T. H., de Melo, P. O. V., Almeida, J. M., Viana, A. C., Salles, J., and Loureiro, A. A. (2014b). Definição, modelagem e aplicações de camadas de sensoriamento participativo. In *Proc. of XXXIII SBRC'14*, Florianópolis, SC.

Silva, T. H., Vaz de Melo, P. O. S., Almeida, J. M., Borges Neto, J., Tostes, A. I. J., Celes, C. S. F. S., Mota, V. F. S., Cunha, F. D., Ferreira, A. P. G., Machado, K. L. S., and Loureiro, A. A. F. (2015). Redes de sensoriamento participativo: Desafios e oportunidades. In *Magnos Martinello; Moises Renato Nunes Robeiro; Antônio Augusto Aragão Rocha. (Org.). Minicursos / XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 266–315, Porto Alegre. SBC.

Silva, T. H., Vaz de Melo, P. O. S., Almeida, J. M., and Loureiro, A. A. F. (2013). Uma Fotografia do Instagram: Caracterização e Aplicação. In *Proc. of XXXII SBRC'13*, Brasília, DF.

Tostes, A. I. J., de LP Duarte-Figueiredo, F., Assunção, R., Salles, J., and Loureiro, A. A. (2013). From data to knowledge: city-wide traffic flows analysis and prediction using bing maps. In *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing*, page 12. ACM.

DSVis: Ferramenta para Edição e Visualização de Rastros de Execução de Sistemas Distribuídos

Túlio Gomes Barbosa[1], **Daniel Thales Naves**[2]
Luis Fernando Faina[3], **Lasaro Camargos**[3]

[1] Serpro, Belo Horizonte-MG

[2]TQI Consultoria, Uberlândia-MG

[3]Universidade Federal de Uberlândia, Uberlândia-MG *

Resumo. *Visualização é um poderoso aliado no desenvolvimento e entendimento de sistemas e algoritmos. Em sistemas distribuídos, por exemplo, a visualização da troca de mensagens, mesmo independentemente dos estados dos diversos processos, pode fornecer insights como a identificação de gargalos na comunicação. No estudo de algoritmos distribuídos, a visualização de exemplos cuidadosamente criados facilita o entendimento. Neste trabalho é apresentado um sistema para edição e visualização de rastros de comunicação de sistemas distribuídos, que tem como objetivo facilitar o estudo e desenvolvimento destes sistemas. Rastros de comunicação são escritos em uma linguagem formal e extensível, facilmente gerada manualmente ou por aplicações. O sistema interpreta rastros e gera visualizações animadas e estáticas. Aqui são apresentadas duas visualizações, diagrama espaço×tempoe árvore hiperbólica. Contudo, o sistema pode ser visto como uma infraestrutura facilmente extensível, a qual novas visualizações podem ser adicionadas.*
Palavras-chave*: sistema distribuído; algoritmo distribuído; rastro de comunicação; diagrama espaço×tempo; árvore hiperbólica.*

Abstract. *Visualization is a powerful tool in developing and understanding systems and algorithms. In distributed systems, for example, visualizing communication patterns, even without considering the internal state of processes, may lead to insights such as communication bottlenecks and emergent behavior. Carefully crafted visualizations of examples also facilitate the study of algorithms. In this work we present a framework for visualization of distributed systems communication traces, with the aim of facilitating the study and development of algorithms. Communication traces are written in simple and extensible formal language, and can be easily crafted manually. Our system interprets such traces and generates animated and static visualizations. Here we present two visualizations, space×time diagrams and hyperbolic trees. However, other visualizations can be easily added to the system.*
Key-words: *distributed system; distributed algorithm; communication trace; visualization; space×time diagram; hyperbolic tree.*

*Os autores agradecem à Fapemig, CAPES e CNPq pelo suporte a este trabalho.

1. Descrição do Problema

Um Sistema Distribuído é um sistema computacional cujos componentes se coordenam e cooperam via troca de mensagens sobre uma rede de comunicação [Coulouris et al. 2011]. Estes componentes poder ser implementados em *hardware* ou *software*, mas neste contexto, podemos nominar os componentes apenas como "processos".

Esta descrição genérica é adequada para muitos sistemas, com propósitos diversos, como mecanismos de pesquisa e indexação da *web*, jogos *online multiplayer*, sistemas de controle de versão distribuídos, etc. Em função das diferentes peculiaridades e complexidades, desenvolvê-los ou entendê-los demanda considerável esforço.

Técnicas de visualização de conceitos abstratos são ferramentas importantes para o estudo e compreensão de algoritmos e estruturas de dados, como p.ex., [Visualgo 2016]. O mesmo é verdade na computação distribuída, o que é facilmente comprovado em livros texto e artigos da área que usam, por exemplo, diagramas espaço×tempo para descrever execuções típicas e atípicas de algoritmos distribuídos.

Em um diagrama espaço×tempo as linhas horizontais representam o tempo, as linhas verticais representam o espaço (processos neste contexto), os pontos representam eventos como envio e recepção de mensagens, e as setas entre eventos representam troca de mensagens [Wu 1998]. A Figura 1 exibe um exemplo de diagrama espaço×tempo.

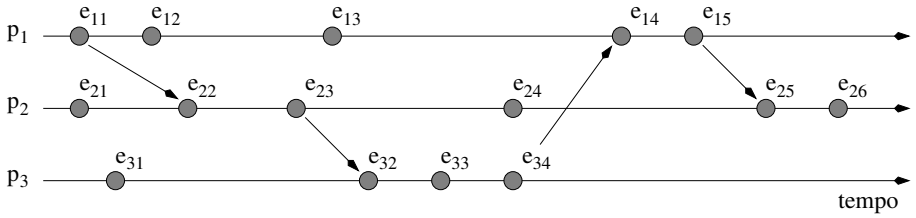


Figura 1. Exemplo de diagrama espaço tempo com três processos.

Diferentes visualizações têm diferentes benefícios. Um diagrama de mesmo propósito é a árvore hiperbólica, que explicita a comunicação do ponto de vista de um processo. Outro interessante é o diagrama de cordas, que permite comparar o número de mensagens trocadas entre processos e identificar gargalos.

Neste trabalho, propõe-se a criação de um *framework* para a visualização de rastros de comunicação de sistemas distribuídos como ferramenta para o seu estudo e desenvolvimento. Neste contexto, descreve-se a arquitetura proposta, aspectos da implementação do protótipo, a linguagem formal de especificação de rastros e duas visualizações: diagrama espaço×tempo e árvore hiperbólica. Para demonstrar as diversas funcionalidades da ferramenta, são também apresentados alguns exemplos.

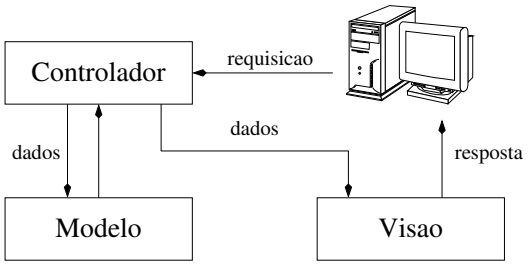
2. DSVis – Ferramenta para Edição e Visualização de Rastros de Comunicação de Sistemas Distribuídos

Diversas nuances são normalmente usadas para descrever o modelo computacional de um sistema distribuído, p.ex., modelo de falha, comunicação e sincronismo. Neste trabalho tais nuances são relevantes apenas em como afetam a geração de rastros de comunicação. Nestes modelos, destacam-se as seguintes características:

- processos tem acesso a um relógio local usado para marcar os eventos de transmissão e recepção de mensagens e não se assume como premissa a sincronização entre os relógios locais. Não obstante, a falta de sincronização pode levar a interpretações diferentes dos eventos por diferentes processos;
- mensagens podem ser perdidas em trânsito, gerando um evento de envio de mensagem sem correspondente entrega. No rastro de execução, pre-processado antes da visualização, mensagens perdidas são denotadas por tipo de mensagem específico e com estimativa de tempo de entrega esperado;
- mensagens não são duplicadas ou os eventos de entrega de uma duplicata são silenciosamente ignorados e não adicionados ao rastro;
- falha de um processo é um evento instantâneo em sua execução, que não o obriga nem o inibe de enviar mensagens em instantes posteriores.

2.1. Arquitetura Proposta

O VSDis foi construído segundo a Arquitetura MVC (*Model, View, Controller*), onde os rastros constituem o modelo, o interpretador dos rastros é o controlador e as várias visualizações constituem a visão (Figura 2(a)).




```
1  var v = new view.SpaceTimeDraw(...);
2  //var v = new view.Hypervis(...);
3  var c = new controller.
       DiagramController(diagram, v);
4  c.updateView();
```

Figura 2. a) Modelo MVC. b) MVC no SVDis.

O uso desta arquitetura torna o *framework* facilmente extensível. Por exemplo e de forma simplificada, trocar a visão de diagrama espaço×tempo para árvore hiperbólica compreende comentar a linha 1 e descomentar a linha 2 no pseudo-código da Figura 2(b).

2.2. Tecnologias Utilizadas

Dentre as tecnologias empregadas no DSVis, destacam-se as seguintes.

Javascript A linguagem JavaScript foi escolhida principalmente por permitir o uso *online* da ferramenta, diretamente de um navegador e sem a necessidade de instalação.

Jison Jison[1] é uma ferramenta que recebe como entrada uma gramática livre de contexto e gera como saída um *parser* JavaScript da linguagem especificada. O *parser* transforma sua entrada em código JavaScript executável por um interpretador. No DSVis, a linguagem especifica o formato do rastro e o *parser* gerado pelo Jison transforma tais rastros em código JavaScript que gera as visualizações. Por exemplo, sempre que encontra a declaração de uma nova linha, o *parser* gera código para adicionar os processos de origem e destino ao conjunto de processos, caso ainda não estejam lá.

Browserify O código foi modularizado usando o *Browserify*, que integra código separado em diversos arquivos e também o "minifica". Isto é, comprime o resultado para minimizar tempo de transferência na rede para o navegador.

[1] `http://zaa.ch/jison/docs/`

JQuery e CSS Na construção da interface gráfica do DSVis também foram usados o JQuery, que facilita a construção e manipulação de formulários, inclusive edição de rastros de execução e, CSS (Cascade Styling Sheets), que separa o estilo das *tags* de suas declarações, etc.

2.3. Funcionalidades Contempladas no DSVis

Nesta seção descrevem-se algumas das funcionalidades do DSVis. A maior parte delas está disponível apenas para diagramas espaço×tempo.

2.3.1. Rastro Amigável a Humanos

A linguagem usada nos rastros é simples e intuitiva, podendo ser gerada manualmente. Sua gramática está disponível no repositório do projeto[2]. De forma simplificada, cada entrada ou linha do rastro segue o formato

```
1 *src sLbl|null sTime -->|..>|--x|..x dst rLbl|null rTime: mLbl --color c --weight w
```

onde os campos tem o seguintes significados

- src: processo que envia a mensagem;
- sLbl: texto descritivo do evento de envio da mensagem;
- sTime: momento em que a mensagem foi enviada;
- `-->|..>|--x|..x`: tipos de traço e terminação da mensagem enviada;
- dst: processo que recebe a mensagem;
- rTime: momento em que a mensagem foi recebida;
- rLbl: texto descritivo do evento de recepção da mensagem;
- mLbl: texto descritivo do conteúdo da mensagem;
- –color c: parâmetro opcional para a cor da linha que representa a mensagem (valor hexadecimal RGB, se omitido, cor padrão é preto);
- –weight w: parâmetro opcional para o peso da linha que representa a mensagem (se omitido - peso padrão é 1).

2.3.2. Tipos de Mensagens

Quatro tipos de mensagens são possíveis, descritas a seguir e demonstradas na Figura 3:

- `-->` linha contínua entregue com sucesso ao destinatário;
- `..>` linha tracejada entregue com sucesso ao destinatário;
- `--x` linha contínua não entregue ao destinatário; e,
- `..x` linha tracejada não entregue ao destinatário;

Na atual versão do DSVis, mensagens não entregues devem ser explicitadas no rastro, gerado por exemplo por uma ferramenta que faça um pré-processamento de rastros reais. O processo será simplificado no futuro.

[2] https://github.com/pluxos/dis-sys-vis/blob/master/diagrams-lib/grammar.jison

```
1 A p1 5-->B p2 8:req
2 B p3 9--xA p4 12:lost resp --color #881A1B
3 A p1 15-->B p2 20:req
4 B p3 21--xA p4 26:resp --color #00FF00
```

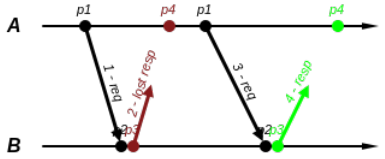


Figura 3. Tipos de mensagens representadas dentro do diagrama espaço tempo.

2.3.3. Processos Invisíveis

Para simplificar a visualização e explicitar somente a comunicação, é possível ocultar a linha do tempo dos processos, tornando-os "invisíveis". Isto é feito adicionando-se um * no início do nome do processo a ser ocultado, como o processo `cliente` na Figura 4.

```
1 *A p1 5-->A p3 15:req
2 A p1 5-->B p2 10:req
3 *B p1 10-->A p2 15:req
```

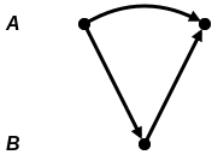


Figura 4. Processos invisíveis e mensagens em *loopback*

2.3.4. Mensagens em *loopback*

Uma mensagem que tem como origem e destino o mesmo processo é denominada *loopback* e também é suportada no DSVis. Para mensagens dessa natureza, não são usadas linhas retas e sim traços no formato de curvas, o que facilita a visualização e identificação das mensagens. Figura 4 exemplifica mensagens em *looback*.

2.3.5. Mensagens não ordenadas

É possível processar mensagens não ordenadas no DSVis. Nesse caso, as mensagens são exibidas em ordem temporal, independente da sequência de processamento do rastro. A linha 4 da figura 5 exemplifica esse tipo de mensagem.

```
1 B p1 10-->C p2 20:request
2 B p3 15-->C p4 23:request
3 C p5 25..>B p6 30:lost request --color #881
    A1B
4 B s1 5..>C s2 15:lost request --color #
    FF8D00
```

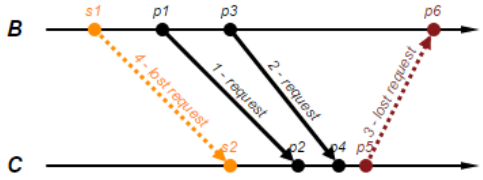


Figura 5. Troca de mensagens não ordenadas

2.3.6. Mensagens coloridas

Outra funcionalidade que facilita a visualização e identificação de mensagens é opção de adicionar cores usando seu código RGB (Figura 3).

2.3.7. Níveis de Detalhamento

A ferramenta contempla uma série de *checkboxes*, ilustrados na Figura 6, que permitem personalizar a saída padrão dos diagramas gerados da seguinte forma:

- ***Label***: exibe nomes do remetente e receptor junto aos eventos;
- ***Time***: exibe os tempos em que os eventos ocorreram;
- ***Contents***: exibe o conteúdo da mensagem junto à aresta que a representa;
- ***Animated***: anima o processo de desenho das linhas e das mensagens, respeitando a ordem especificada no rastro.

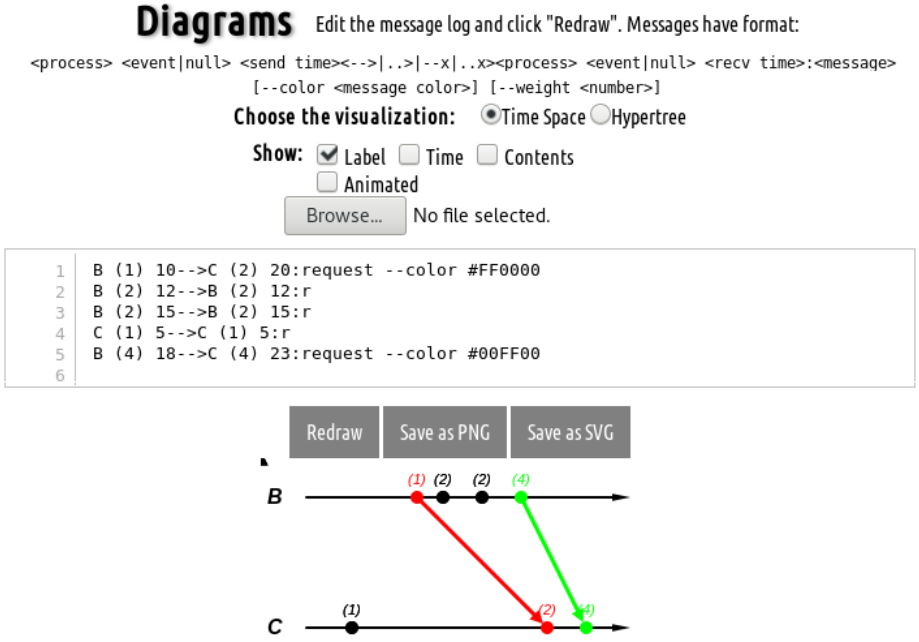


Figura 6. Opções para realização do desenho do diagrama.

2.3.8. Salvar e Carregar

A aplicação dispõe de três botões, pelos quais é possível reiniciar a visualização (*Redraw*) e salvar a visualização em formato **PNG** ou **SVG**. Os botões são vistos na Figura 6 imediatamente acima do diagrama espaço×tempo.

3. Código Fonte, Instalação e Execução

O código fonte do DSVis está disponível no sítio do projeto, `http://pluxos.github.io/dis-sys-vis/` e é distribuído sob Licença BSD. Embora novas funcionalidades estejam sendo adicionadas no DSVis constantemente, o mesmo já é usável em duas formas, online ou localmente. Para uso online, basta acessar `http://pluxos.github.io/dis-sys-vis/live` com algum navegador moderno.

Para acesso local, é necessário executar procedimento de instalação detalhando no sítio do projeto `http://pluxos.github.io/dis-sys-vis/`. Em essência,

basta instalar o *Node.js* e o *Browserify*, clonar o projeto DSVis, e executar o comando comando `make all` a partir da pasta do projeto. Assim gera-se um arquivo JavaScript único a ser incluso em uma página HTML, especificando um rastro de execução a ser visualizado. A página do demo online está presente na pasta live/index.html e pode ser usada de base para a criação de aplicações mais completas.

4. Manuais

O DSVis tem uso intuitivo, então é possível usá-lo simplesmente consultado-se os exemplos disponíveis com o projeto. Ainda assim, um manual simples está disponível na página principal do projeto.

5. Trabalhos relacionados

Há diversos trabalhos relacionados à visualização de rastros em sistemas distribuídos. A seguir revisamos os dois que serviram de inspiração para o VSDis.

5.1. Lupa

O Projeto Lupa [Garcia 2005] provê um ambiente de visualização e animação de algoritmos de *checkpointing* distribuído. A entrada do Lupa é um algoritmo e a saída um diagrama espaço×tempo, gerado pela execução do algoritmo. Comparado ao VSDis, o Lupa é mais restrito por considerar apenas algoritmos de *checkpointing* e mais geral por gerar ele mesmo os rastros a serem visualizados.

5.2. JS-Sequence-Diagram

A ferramenta *js-sequence-diagrams*, desenvolvida em JavaScript, gera diagramas de sequência a partir de entradas textuais de acordo com uma linguagem especificada em Extended Backus-Naur Form, EBNF. Comparado ao DSVis, o *js-sequence-diagrams* é mais limitado pois foi desenhado para gerar diagramas de sequência.

6. Demonstração

A demonstração usará de vários exemplos disponíveis para ilustrar as funcionalidades descritas anteriormente.

Uma vez que a ferramenta pode ser executada online, para a demonstração no SBRC serão necessários apenas um computador com navegador moderno e conexão com a Internet. Se não for possível prover a conexão, a demonstração poderá ser executada à partir do próprio computador.

A tela principal que é apresentada para o usuário pode ser visualizada na Figura 6, um exemplo básico é carregado como padrão com o intuito de facilitar a criação de novos diagramas, bastando seguir o modelo proposto.

7. Conclusão e Trabalhos Futuros

O objetivo do trabalho foi a construção de uma ferramenta para edição e visualização de rastros de comunicação de sistemas distribuídos. Para isso, foi construída uma biblioteca que analisa rastros e invoca visualizações escolhidas. Foram implementadas duas visualizações e criada uma biblioteca de exemplos, invocáveis de uma aplicação com

editor de rastros que facilita a geração de diagramas em "tempo real". A ferramenta é facilmente extensível pela modificação da linguagem e adição de novas visualizações.

Diversas funcionalidades estão presentes na geração das visualizações, como entrada de dados textuais amigável à humanos, mensagens em *loopback*, uso de cores, exportação dos diagramas utilizando *PNG* e *SVG*, etc.

A ferramenta, apesar de ainda em desenvolvimento, tem sido usada por um dos autores em suas aulas de Sistemas Distribuídos para exemplificação interativa execuções de algoritmos. O resultados iniciais são bons. O código da mesma é livre sob licença BSD e disponível publicamente a partir do sítio do projeto, `http://pluxos.github.io/dis-sys-vis/`.

Diversas novas funcionalidades estão planejadas, como interpretação de relógios lógicos de Lamport e Vetoriais no estado dos processos, bem como sua geração automática a partir dos eventos contidos nos rastros. Também está sendo desenvolvida uma visualização em cordas.

Referências

Coulouris, G., Dollimore, J., Kindberg, T., and Blair, G. (2011). *Distributed Systems: Concepts and Design*. Addison-Wesley Publishing Company, USA, 5th edition.

Garcia, I. C. (2005). Ambiente para animação de algoritmos distribuídos baseado em construções progressivas de checkpoints globais consistentes. http://www.bv.fapesp.br/pt/bolsas/61527/ambiente-para-animacao-de-algoritmos-distribuidos-baseado-em-construcoes-progressivas-de-checkpoints/. Acessado em: 16/03/2016.

Visualgo (2016). visualgo. https://visualgo.net/. Acessado em: 03/03/2017.

Wu, J. (1998). *Distributed System Design*. Taylor & Francis.

Salão de Ferramentas do SBRC 2017
Sessão Técnica 4
Redes sem Fio e IoT II

SensingBus: um Sistema de Sensoriamento Baseado em Ônibus Urbanos

Pedro Cruz[1], Felipe F. da Silva[1], Roberto G. Pacheco[2], Rodrigo S. Couto[2], Pedro B. Velloso, Miguel Elias M. Campista[1] e Luís Henrique M. K. Costa[1]

[1]Universidade Federal do Rio de Janeiro - PEE/COPPE/GTA
[2]Universidade do Estado do Rio de Janeiro - PEL/DETEL/FEN

{cruz,felipe,velloso,miguel,luish}@gta.ufrj.br
{roberto.pacheco,rodrigo.couto}@uerj.br

Abstract. *To develop Smart City applications, urban Internet of Things infrastructures are fundamental to sensing and communication tasks. The trivial alternative is to deploy static Wireless Sensor Networks (WSNs), but this becomes prohibitive as the number of sensors grows. This paper proposes SensingBus, a general-purpose system to collect data with sensors embedded into urban buses. SensingBus benefits from nodes' mobility to enlarge the monitored area. SensingBus follows a three-layer architecture: one collects data; another receives and pre-processes data before sending it over Internet; and a third one stores, processes, and delivers data to end users. Results show SensingBus provides sampling and transmission rates suitable to typical Smart City applications.*

Resumo. *Para desenvolver aplicações de Cidades Inteligentes, infraestruturas urbanas de Internet das Coisas são fundamentais em tarefas de sensoriamento e comunicação. A alternativa trivial, com a instalação de uma Rede de Sensores Sem Fio (RSSF) estáticos, pode se tornar proibitiva com o aumento do número de nós. Este trabalho propõe o SensingBus, um sistema que provê mobilidade a sensores ao embarcá-los em ônibus urbanos, expandindo a área monitorada. A arquitetura adotada possui três camadas: uma coleta dados; outra pré-processa dados antes do envio pela Internet; e uma terceira armazena, processa e entrega dados a usuários finais. Resultados mostram que o SensingBus oferece taxas de amostragem e transmissão compatíveis com aplicações de Cidades Inteligentes.*

1. Introdução

Cidades Inteligentes utilizam dados obtidos através de diferentes métodos com a finalidade de aprimorar os serviços fornecidos aos cidadãos, além de possibilitar a oferta de novos serviços. Muitos desses dados podem ser coletados através do paradigma da Internet das Coisas (*Internet of Things* - IoT), no qual objetos cotidianos são dotados de interfaces de comunicação, processamento e sensoriamento [Gubbi et al., 2013].

O sensoriamento em Cidades Inteligentes requer o espalhamento de sensores por uma grande região geográfica, o que pode apresentar um custo proibitivo. Uma opção de contorno é aplicar uma Rede de Sensores Sem-Fio Móveis cobrindo a região, utilizando a mobilidade desses sensores para reduzir a infraestrutura necessária. Adicionalmente, é possível empregar uma infraestrutura de Computação em Nuvem para processar e armazenar os dados coletados por diferentes dispositivos, a fim de compensar as limitações

dos dispositivos da IoT. As ações de entrada (p.ex., sensores) e saída (p.ex., atuadores) continuam sendo realizadas pelos dispositivos, enquanto as tarefas de processamento são realizadas na nuvem, dotada de maior capacidade de processamento. Uma preocupação com esta configuração se refere ao tráfego trocado na rede, que pode ser elevado. Outra preocupação é a segurança, já que os dispositivos de IoT possuem recursos limitados, e nem sempre são capazes de implementar protocolos seguros para comunicação com a nuvem. Essa limitação pode ser compensada através de uma infraestrutura, denominada névoa, capaz de pré-processar os dados antes que os mesmos trafeguem pela Internet [Coutinho et al., 2016]. Assim, é possível empregar protocolos de segurança mais sofisticados, além de as mensagens poderem ser agregadas ou filtradas na névoa, reduzindo o tráfego com a nuvem.

A solução de sensoriamento adotada pelos projetos Mosaic [Dong et al., 2015], Opensense [Marjovi et al., 2015] e BusNet [Zoysa et al., 2007] é a de utilizar veículos do transporte público para coletar dados de qualidade do ar ou de condição de vias das cidades e oferecê-los a aplicações de usuários. Além de dotar os sensores de mobilidade sem custo adicional, o uso dos ônibus urbanos possui a vantagem dos mesmos seguirem trajetórias pré-determinadas, dando previsibilidade às regiões sensoreadas por cada nó. Os projetos Opensense e Busnet não utilizam uma infraestrutura de nuvem para armazenar e processar os dados coletados. O projeto Mosaic utiliza uma infraestrutura de nuvem para armazenar e processar os dados coletados, mas não realiza nenhum pré-processamento antes dos dados serem enviados pelos nós de sensoriamento à nuvem.

Este trabalho apresenta o SensingBus, um sistema para coleta de dados ambientais no contexto de Cidades Inteligentes. Dentro do paradigma de IoT, os ônibus de transporte público são utilizados para prover mobilidade aos nós de uma rede de sensores sem-fio. Diferentemente dos projetos citados anteriormente, o SensingBus utiliza uma arquitetura de três camadas proposta por [Li et al., 2016]. Nessa arquitetura, os dispositivos IoT realizam a coleta de dados em uma ponta e uma infraestrutura de nuvem executa o processamento e armazenamento dos dados coletados na outra ponta. Adicionalmente, é utilizada uma infraestrutura de névoa entre os dispositivos IoT e a nuvem, para reduzir problemas de desempenho e segurança causados pelo tráfego dos dados pela Internet. A camada de névoa permite que o sistema SensingBus possa autorizar e filtrar os dados que chegam aos usuários sem que sejam necessários dispositivos de sensoriamento com poder computacional elevado. A análise do desempenho do SensingBus mostra que é possível alcançar uma taxa de amostragem de aproximadamente 1 medição por segundo, o que, segundo [Zanella et al., 2014], é o suficiente para aplicações de controle de congestionamento, monitoramento da qualidade do ar, ou ainda iluminação inteligente.

Este trabalho está organizado da seguinte forma. A Seção 2 descreve as funcionalidades do sistema SensingBus. A Seção 3 apresenta sua arquitetura, detalhando as funções e a implementação de cada camada. A Seção 4 apresenta uma análise do desempenho do SensingBus, enquanto a Seção 5 descreve como será realizada a demonstração. Por fim, a Seção 6 conclui o artigo e aponta trabalhos futuros.

2. Funcionalidades do SensingBus

O SensingBus é um sistema completo que implementa desde as principais funcionalidades de coleta de dados ambientais em uma cidade, até o armazenamento e a

entrega dos dados para os usuários. O sistema tem por objetivo permitir que serviços de manutenção e controle da cidade sejam aperfeiçoados, através do mapeamento de suas condições em tempo real. De maneira resumida, as principais funcionalidades do SensingBus são:

- ***Coleta de dados ambientais*:** o SensingBus realiza a coleta de dados ambientais ao longo dos trajetos dos ônibus que possuem nós de sensoriamento embarcados.
- ***Pré-processamento dos dados*:** o SensingBus executa um pré-processamento dos dados antes de transmiti-los pela Internet, eliminando dados inconsistentes e, consequentemente, reduzindo a quantidade de dados que trafegam pela Internet. Adicionalmente, através de chaves e certificados, o sistema impede que dispositivos não autorizados publiquem dados ou alterem dados legítimos antes que esses cheguem aos usuários.
- ***Busca de dados sensoreados*:** usuários podem buscar dados que estejam na base de dados, através de uma API, e recebê-los no formato JSON. Os dados podem ser filtrados por janela de tempo, coordenadas geográficas, identificador do nó de Coleta ou pelo ônibus que realizou a coleta.
- ***Armazenamento de dados sensoreados*:** dispositivos autorizados podem inserir dados no servidor da estrutura de Computação em Nuvem, através de uma API. Assim, é possível a integração do sistema com outras fontes de dados.
- ***Visualização de dados sensoreados*:** usuários podem buscar dados que estejam na base de dados e visualizá-los em um mapa. Os dados a serem visualizados no mapa também podem ser filtrados por tipo de dado, janela de tempo, coordenadas geográficas ou pelo ônibus que realizou a coleta. A visualização é feita através de um navegador, sem a necessidade de instalações de software por parte do usuário.

A implementação do sistema utiliza protocolos abertos e é baseada em uma arquitetura em camadas, apresentada na Seção 3. Estas características privilegiam a modularidade e extensibilidade do sistema.

3. Arquitetura do SensingBus

A arquitetura do SensingBus, ilustrada na Figura 1, é dividida em três camadas: Coleta, Recepção e Publicação. Os dados são obtidos pela camada de Coleta, que é composta por nós de sensoriamento situados nos veículos. Estes dados são então repassados para a camada de Recepção, que consiste em nós fixos instalados pela cidade, como em pontos de ônibus. Essa camada realiza pré-processamento dos dados, atuando como o que se convencionou chamar de névoa computacional, além de garantir a autenticidade dos dados. A camada de Publicação, situada na nuvem, recebe e processa os dados dos nós da Recepção, realizando a maior parte do processamento na arquitetura do SensingBus. Por fim, essa camada disponibiliza os dados aos usuários. Cada uma das camadas do SensingBus é detalhada a seguir.

3.1. Coleta

A camada de Coleta é responsável por colher dados sobre a cidade e entregá-los à camada de recepção. Antes da entrega, os dados são associados ao local e momento em que foram coletados. Essa camada é composta de nós de Coleta acoplados aos ônibus que circulam pela cidade. A arquitetura de cada nó de coleta, ilustrada na Figura 1, segue a arquitetura básica de dispositivos IoT [Santos et al., 2016].

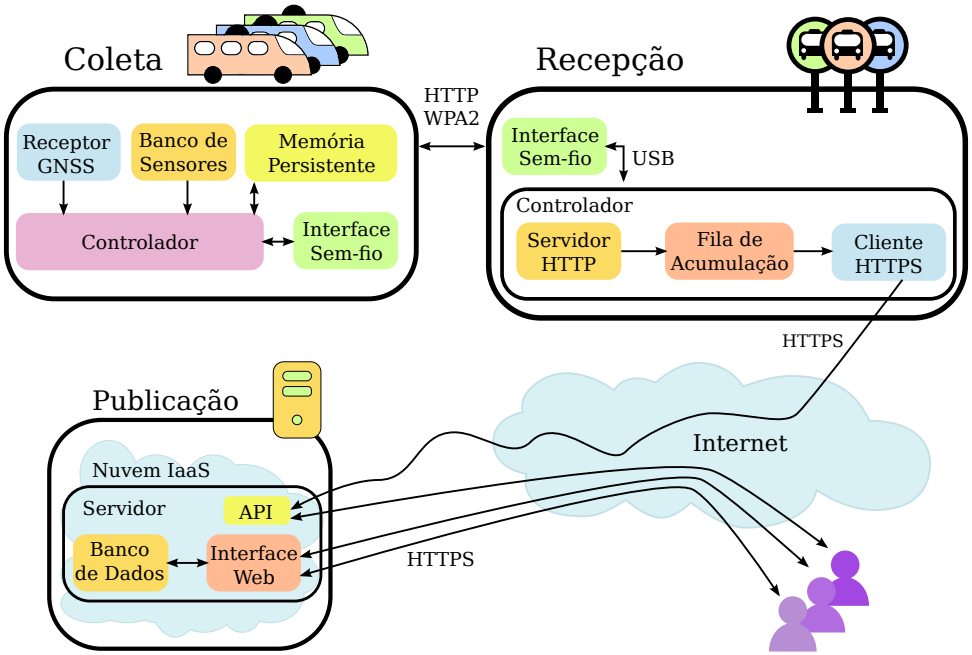


Figura 1. Arquitetura do sistema SensingBus: nós de Coleta, Recepção e Publicação.

Dentro da Coleta, um Controlador gerencia todas as tarefas de cada nó de coleta. A uma determinada taxa de amostragem, o Controlador adquire as medidas coletadas pelos sensores do Banco de Sensores e as associa à posição e ao horário indicados pelo Receptor GNSS (*Global Navigation Satellite System*). Após a associação dos dados coletados à sua posição e momento de coleta, o Controlador armazena os dados na Memória Persistente. Em cada iteração, o Controlador consulta a Interface Sem-Fio sobre a existência de uma conexão com a camada Recepção. A associação entre um nó de Coleta e um nó de Recepção se dá através de WPA2, com uma chave compartilhada entre os dois nós. Se houver conexão, o Controlador envia os dados armazenados na Memória Persistente para a Interface Sem-Fio, que transmite os dados para a Recepção utilizando um POST HTTP.

O Controlador envia para a Interface Sem-Fio apenas os dados gerados pelo Banco de Sensores e pelo Receptor GNSS. A Interface Sem-Fio, por sua vez, se encarrega de encapsular os dados com os cabeçalhos do POST. Essa separação de tarefas é realizada para melhorar o desempenho da comunicação, visto que o Controlador possui baixo poder de processamento. Os resultados apresentados na Seção 4 mostram que essa abordagem aumenta em mais de três vezes a vazão da comunicação entre a Coleta e a Recepção, em comparação com o cenário no qual o Controlador realizaria o encapsulamento HTTP.

Os equipamentos utilizados para implementar o nó de Coleta estão relacionados na Tabela 1. O Arduino UNO R3 foi escolhido como Controlador por ser um equipamento de baixo custo com poder computacional suficiente para executar as tarefas do Controlador. O nó de Coleta é instalado na parte interna do ônibus, exceto pelo Banco de Sensores e a Interface Sem Fio, que são instalados na parte externa do ônibus dentro de uma caixa protetora.

Tabela 1. Equipamentos componentes do nó de Coleta.

Módulo	Equipamento	Fabricante
Controlador	Arduino UNO R3	Arduino
Receptor GNSS	GS-96U7	Guangzhou Xintu
Memória Persistente	GS-96U7	Guangzhou Xintu
Interface sem Fio	ESP8266	Espressif
Banco de Sensores	Sensor de Umidade DHT11	DFRobot
	Sensor de Temperatura DHT11	DFRobot
	Sensor de Intensidade Luminosa GL5528	GBK Robotics
	Sensor de Intensidade de Chuva GL5528	GBK Robotics

3.2. Recepção

Os nós da camada Recepção são responsáveis por receber os dados brutos da Coleta, analisá-los e filtrar possíveis inconsistências. Essas inconsistências podem ocorrer devido a erros de associação entre os dados ambientais e os dados de localização obtidos pela Coleta, ou até mesmo por erros dos sensores durante a coleta de dados. As inconsistências são detectadas quando os dados estão fora de uma faixa considerada viável.

A Recepção utiliza o conceito de computação em névoa (ou *fog computing*) [Coutinho et al., 2016]. Nessa arquitetura, parte dos recursos computacionais são deslocados para a borda da rede, a fim de realizar o pré-processamento dos dados da Coleta. Um exemplo de pré-processamento oferecido pela Recepção trata-se da função da Fila de Acumulação de dados. A Fila de Acumulação atua como um *buffer*, armazenando os dados recebidos pela Recepção durante um determinado período de tempo. Após isso, agrega todos esses dados em uma mensagem e os envia à Publicação. Dessa maneira, a Fila de Acumulação reduz a sobrecarga de cabeçalhos, diminuindo o tráfego entre essas camadas. No SensingBus, o conceito de névoa também é usado para possibilitar que apenas nós da Coleta autorizados consigam enviar dados para a Publicação. Para tal, cada nó da Recepção implementa um ponto de acesso IEEE 802.11, que oferece uma rede privada baseada no padrão WPA2 (*Wi-Fi Protected Access 2*). Assim, os nós da Coleta se conectam a essa rede utilizando senhas pré-configuradas. As mensagens entre a Coleta e a Recepção são enviadas por HTTP.

Em relação à Publicação, cada nó da Recepção comporta-se como um cliente HTTPS, enviando dados para o servidor situado na Publicação, como ilustrado na Figura 1. O uso do HTTPS possibilita a existência de comunicação segura entre a Recepção e a Publicação. Além da verificação do certificado do servidor da Publicação, intrínseca ao protocolo HTTPS, a Recepção também apresenta um certificado a esse servidor. Esse certificado é assinado por uma Autoridade Certificadora reconhecida pela Publicação. Essa medida visa assegurar que os nós da Publicação só aceitem dados provenientes de nós autorizados.

Os equipamentos utilizados para implementar a camada Recepção neste trabalho estão relacionados na Tabela 2. A escolha do Raspberry Pi como o equipamento da Recepção deve-se, sobretudo, ao seu baixo custo e sua configuração de *hardware* satisfatória para implementar as funcionalidades desejadas. Assim, para um orçamento

limitado, mais pontos de ônibus podem ser utilizados na arquitetura.

Tabela 2. Equipamentos componentes do nó de Recepção.

Módulo	Equipamento	Fabricante
Controlador	Raspberry Pi II model B	Raspberry Pi Foundation
Interface Sem-Fio	TL-WN722N	TP-LINK

3.3. Publicação

No SensingBus, a Publicação é o destino dos dados obtidos por todos os nós da Coleta, após o pré-processamento pela Recepção. Os dados são armazenados e ficam disponíveis para os usuários. O nó de Publicação consiste em uma máquina virtual hospedada em uma nuvem geodistribuída, fornecida pelo projeto PID [Couto et al., 2015]. A nuvem utilizada é do tipo IaaS (*Infrastructure as a Service* - Infraestrutura como Serviço), implementada através do orquestrador Openstack[1]. Na Publicação, um servidor Apache[2] recebe requisições HTTPS, que são tratadas pela aplicação Django[3]. Os dados são armazenados e consultados no banco de dados relacional MySQL[4].

A Publicação oferece uma URL para inserção de dados, que é protegida pelo servidor Apache. Apenas clientes que apresentem um certificado reconhecido pelo servidor são autorizados a escrever na base de dados. Isso implica que, além dos dados e metadados gerados pela Coleta, os POSTs de um nó de Recepção devem conter um certificado gerado pela Autoridade Certificadora na qual o servidor possui confiança, mencionado na Seção 3.2. Dessa forma, os privilégios de escrita são fornecidos apenas aos dispositivos que possuem certificados assinados pela Autoridade Certificadora. Esse procedimento visa impedir que o nó da Publicação receba dados falsos, gerados por dispositivos maliciosos se passando por nós de Recepção. Assim, a segurança do SensingBus é garantida pelo WPA2, entre a Recepção e a Coleta, e por chaves assimétricas e certificados, entre a Recepção e a Publicação.

Usuários que busquem dados na Publicação não precisam ser autorizados, pois é considerado que os dados coletados pelo SensingBus são públicos. Durante a consulta de dados à Publicação, tanto por API quanto pela interface *web*, é possível que os usuários apliquem filtros de busca, como mencionado na Seção 2. Pela interface *web*, os dados visualizados são distribuídos em um mapa geográfico da cidade, enquanto que através da API os dados são retornados no formato JSON.

4. Análise do SensingBus

Com a finalidade de verificar as aplicações de sensoriamento que o SensingBus é capaz de atender, nesta seção são descritas medidas de desempenho obtidas com o sistema. Os testes foram realizados em laboratório, a fim de um melhor controle das condições. A Tabela 3 apresenta as medidas realizadas (os valores são acompanhados de intervalo de confiança de 95%).

[1] `https://www.openstack.org/` (acessado em 5 de abril de 2017)
[2] `http://httpd.apache.org/` (acessado em 5 de abril de 2017)
[3] `https://www.djangoproject.com/` (acessado em 5 de abril de 2017)
[4] `https://www.mysql.com/` (acessado em 5 de abril de 2017)

Tabela 3. Análise do SensingBus.

Grandeza	Valor medido
Taxa de amostragem de sensoriamento	1,02 $\pm$ 0,01 Hz
Taxa de transmissão entre nós das camadas de Coleta e de Recepção	1.302,5$\pm$0,9 B/s
Taxa de geração de dados	53,0 $\pm$ 0,5 B/s

A taxa de amostragem de sensoriamento é a taxa com que o Banco de Sensores é lido pelo controlador. A taxa de transmissão entre os nós das camadas de Coleta e de Recepção corresponde à quantidade de dados que podem ser enviados por um nó de Coleta para um nó de Recepção, quando há conexão entre os dois. A taxa de geração de dados corresponde à quantidade de dados gerados por segundo por um único nó de Coleta, com o Banco de Sensores e taxa de amostragem de sensoriamento mostrados neste trabalho. As medições obtidas mostram que o sistema é compatível com as taxas de amostragem e geração de tráfego previstas para as aplicações apontadas por [Zanella et al., 2014], como monitoramento de qualidade do ar, que demanda taxas de amostragem da ordem de 5 minutos. Segundo [Da Silva et al., 2013], os tempos médios de contato entre um ônibus e um ponto de acesso localizado em um ponto de ônibus são de 65s e 36s quando o ônibus para no ponto e quando não para no ponto, respectivamente. Então, pode-se afirmar que, em média, os nós de Coleta podem ficar aproximadamente 13 minutos sem contato com nenhum nó de Recepção e ainda assim entregar todos os dados coletados durante o intervalo.

5. Demonstração para o Salão de Ferramentas

Para a demonstração prevista, a arquitetura completa da Figura 1 será reproduzida, com nós da Coleta e da Recepção disponíveis para o público, emulando o cenário com ônibus e pontos de ônibus. Além disso, os usuários poderão acessar a página do projeto[5] e interagir, em tempo real, com a interface gráfica de coleta de medidas. Os equipamentos relacionados nas Tabelas 1 e 2 serão fornecidos pelos autores e apresentados na demonstração. O nó referente à camada de Recepção estará hospedado na nuvem do projeto PID [Couto et al., 2015]. Será necessária conexão com a Internet via cabo para os nós de Recepção e um ou mais computadores com acesso à Internet para que os usuários possam acessar as páginas do SensingBus.

Os usuários poderão encontrar na página do projeto[5] informações referentes ao SensingBus. As informações disponíveis incluem um manual do usuário explicando a utilização do SensingBus e também um guia de instalação, com intuito de que qualquer usuário seja capaz de reproduzir o sistema. Além disso, as documentações dos códigos poderão ser acessadas através da página[5]. Por último, há um repositório[6], no qual encontram-se todos os códigos do SensingBus e instruções para replicação do sistema.

6. Conclusão

Este artigo apresentou o sistema de sensoriamento móvel SensingBus. O SensingBus utiliza conceitos de Internet das Coisas, computação em névoa e computação em

[5] `https://sensingbus.gta.ufrj.br` (acessado em 5 de abril de 2017)

[6] `https://github.com/pedrocruz/sensing_bus` (acessado em 5 de abril de 2017)

nuvem, com o objetivo de coletar dados ambientais de uma cidade. Os ônibus do transporte público são usados como plataforma de mobilidade para nós de sensoriamento e os dados são disponibilizados para os usuários por uma API ou uma interface web. Com o objetivo de atestar o desempenho do SensingBus, foram realizados testes que comprovaram a viabilidade e aplicabilidade do sensoriamento realizado.

Como trabalhos futuros, pretende-se implementar o sistema na frota de ônibus da cidade universitária da Ilha do Fundão e identificar aplicações que utilizem os dados disponibilizados pelo SensingBus.

Referências

[Coutinho et al., 2016] Coutinho, A. A. T. R., Carneiro, E. O. e Greve, F. G. P. (2016). Computação em névoa: Conceitos, aplicações e desafios. Em *Minicursos do XXXIV SBRC*, p. 266–315.

[Couto et al., 2015] Couto, R. S., Sciammarella, T., Barreto, H. F. S. S. M., Campista, M. E. M., Rubinstein, M. G., Costa, L. H. M. K., Vetter, F. e Marins, A. L. A. (2015). GT-PID: Uma nuvem IaaS universitária geograficamente distribuída. Em *Quinta Conferencia de Directores de Tecnología de Información - TICAL 2015*, p. 1–19. Redclara.

[Da Silva et al., 2013] Da Silva, V. B., Da Silva, F. O., Campista, M. E. M. e Costa, L. H. M. (2013). A trajectory-based approach to improve delivery in drive-thru internet scenarios. Em *Communications Workshops (ICC), 2013 IEEE International Conference on*, p. 489–494. IEEE.

[Dong et al., 2015] Dong, W., Guan, G., Chen, Y., Guo, K. e Gao, Y. (2015). Mosaic: Towards city scale sensing with mobile sensor networks. Em *Parallel and Distributed Systems (ICPADS), 2015 IEEE 21st International Conference on*, p. 29–36. IEEE.

[Gubbi et al., 2013] Gubbi, J., Buyya, R., Marusic, S. e Palaniswami, M. (2013). Internet of things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660.

[Li et al., 2016] Li, W., Santos, I., Delicato, F. C., Pires, P. F., Pirmez, L., Wei, W., Song, H., Zomaya, A. e Khan, S. (2016). System modelling and performance evaluation of a three-tier cloud of things. *Future Generation Computer Systems*.

[Marjovi et al., 2015] Marjovi, A., Arfire, A. e Martinoli, A. (2015). High resolution air pollution maps in urban environments using mobile sensor networks. Em *2015 International Conference on Distributed Computing in Sensor Systems*, p. 11 – 20. IEEE.

[Santos et al., 2016] Santos, B. P., Silva, L. A., Celes, C. S., Borges, J. B., Neto, B. S. P., Vieira, M. A. M., Vieira, L. F. M., Goussevskaia, O. N. e Loureiro, A. A. (2016). Internet das coisas: da teoria à prática. p. 1–50.

[Zanella et al., 2014] Zanella, A., Bui, N., Castellani, A., Vangelista, L. e Zorzi, M. (2014). Internet of things for smart cities. *IEEE Internet of Things Journal*, 1(1):22–32.

[Zoysa et al., 2007] Zoysa, K. D., Keppitiyagama, C., Seneviratne, G. P. e Shihan, W. W. A. T. (2007). A public transport system based sensor network for road surface condition monitoring. Em *NSDR '07 Proceedings of the 2007 workshop on Networked systems for developing regions*. NSDR.

TriploS (*Smart Spectrum Sensing)*: uma ferramenta para análise de ocupação de espectro para redes IEEE 802.11

Alex Monteiro[1], Laura Ribeiro[1], Eduardo Souto[1], Rafael Aschoff[2]

[1]Instituto de Computação - Universidade Federal do Amazonas (UFAM), Manaus/AM, Brasil

[2] Instituto Federal de Pernambuco, Palmares/Recife, Brasil

{alex.monteiro,laura.ribeiro,esouto}@icomp.ufam.edu.br

rafael.roque@palmares.ifpe.edu.br

Abstract. *The IEEE 802.11 standard operates in the 2,4 GHz unlicensed band (ISM - Industrial, Scientific and Medical) which, because it doesn't require a license for its use, may lead to signal interference and degradation of performance when the wireless devices are placed in the same environment. This article presents a tool, called TriploS (Smart Spectrum Sensing), which allows sensing and analyzing the occupation of the spectrum by IEEE 802.11g/n wireless networks standard in the 2.4 GHz frequency band. The results generated demonstrate efficiency in the use of the tool constituting open-source and low cost alternative for 2.4 GHz spectrum analysis.*

Resumo.

O padrão IEEE 802.11 opera na faixa de frequência não licenciada de 2.4 GHz (ISM - unlicensed industrial scientific medical) que por não requerer licença para o seu uso pode conduzir a interferência de sinal e degradação de desempenho quando os dispositivos são colocados em um mesmo ambiente. Este artigo apresenta uma ferramenta, denominada TriploS (Smart Spectrum Sensing) que permite sensoriar e analisar a ocupação do espectro por redes sem fio, pertencentes ao padrão IEEE 802.11g/n na faixa de frequência de 2,4 GHz. Os resultados gerados demonstram eficácia no uso da ferramenta constituindo uma alternativa livre e de baixo custo para análise do espectro 2.4 GHz.

1. Introdução

A utilização de redes sem fio IEEE 802.11 (popularmente conhecidas como redes Wi-Fi) para tráfego de dados torna-se cada vez mais comum em nosso cotidiano. A facilidade de uso, aliada à redução dos custos de seus equipamentos como as interfaces de rede e os Pontos de Acesso (PA), motivou um forte crescimento na adoção de redes Wi-Fi como solução de baixo custo tanto para ambientes domésticos como para ambientes coorporativos.

Essa facilidade de uso e implementação, no entanto, pode levar a perda de desempenho quando as redes não são corretamente configuradas. Usuários domésticos, por exemplo, geralmente utilizam as configurações (canal utilizado, protocolo de segurança, identificador da rede, etc) padrão de fábrica no momento da constituição de uma nova rede Wi-Fi. Essa opção, apesar de agilizar o processo de configuração, pode

levar a saturação de canais do espectro de frequência, uma vez que tais canais podem ser compartilhados por diversos pontos de acesso no mesmo ambiente.

Mais especificamente, o padrão IEEE 802.11, assim como outras tecnologias sem fio (por exemplo, Bluetooth e Zigbee), utiliza o espectro 2,4 GHz pertencente à Banda ISM (*Industrial, Scientific and Medical*) que não necessita de uma licença para a sua utilização. Em outras palavras, não há controle direto sobre a utilização dos canais do espectro de frequência. O padrão IEEE 802.11 especifica até 14 diferentes canais de alocação, entretanto, apenas um subconjunto específico desses canais é utilizado na prática. Redes sem fio IEEE 802.11 apresentam canais sobrepostos e canais não sobrepostos, sendo os canais não sobrepostos (canais 1, 6 e 11) os mais utilizados para uso em pontos de acesso Wi-Fi. Os canais não sobrepostos não apresentam índices de sobreposição de frequências sobre os demais canais, reduzindo a possibilidade de interferências entre diferentes canais nas transmissões [IEEE Std 802.11 et al. 2012]]. Contudo, um intenso compartilhamento desses canais pode levar a uma degradação significativa de desempenho das redes sem fio.

Para contornar esse problema, os operadores de redes têm empregado ferramentas que analisam o espectro IEEE 802.11 como inSSIDer [Metageek 2017] e Wifi Analyzer [Farproc 2017] para coletar informações que permitam indicar qual é o melhor canal para a inserção de um novo AP na rede. Tais ferramentas coletam dados como a força de sinal em dBm (RSSI - *Received Signal Strength Indication*) no tempo, qualidade de link (LQ – *Link Quality*), identificador da rede (SSID - *Service Set Identifier*), o endereço MAC do ponto de acesso, entre outros. Tais informações são utilizadas como entrada para algum esquema de seleção e alocação de canais [Balbi et al. 2012][Gramacho et al. 2013][Bhartia et al. 2016] e [Monteiro et al. 2016].

Ferramentas livres para análise de espectro que permitam coletar dados de forma simples, rápida e bem formatada são raras e escassas. Além disso, geralmente essas ferramentas informam as condições das redes e canais existentes no ambiente, cabendo ao usuário à interpretação desses dados. Neste contexto, este trabalho propõe uma ferramenta simples que visa facilitar e auxiliar usuários domésticos, operadores de rede, e pesquisadores que trabalhem com temas de seleção de canal.

O TriploS (*Smart Spectrum Sensing)* é uma ferramenta que visa contribuir para análise de canais IEEE 802.11 para pontos de acesso, de forma a escolher o melhor canal considerando as características de espectro em um determinando ambiente. O TriploS possibilita a coleta de dados e a integração com outras ferramentas para geração de relatórios comportamentais de espectro, exibindo informações como o índice de qualidade de sinal, taxa de dados (*bit rate*) e frequências de operação utilizadas atualmente no ambiente. O TriploS foi desenvolvido para plataforma Linux e dispõe de um código fonte aberto com distribuição gratuita, inclusive interoperando com outras ferramentas de geração de gráficos como o Gnuplot [Gnuplot 2017] e ferramentas de análise comportamental da rede como o Iperf [Dugan et al. 2017].

O restante deste artigo está organizado como segue: A Seção 2 descreve alguns trabalhos que utilizam ferramentas de análise e varredura do espectro IEEE 802.11. A Seção 3 apresenta a arquitetura da ferramenta TriploS. A Seção 4 fornece alguns detalhes operacionais de sua implementação. Na Seção 5 são demonstrados alguns testes avaliativos de funcionamento e desempenho da ferramenta. A Seção 6 descreve como serão realizadas as demonstrações no salão de ferramentas. Por fim, a Seção 7 apresenta as considerações finais sobre o uso da ferramenta e sugestões para trabalhos futuros.

2. Trabalhos Relacionados

Na literatura, a análise e definição de canais IEEE 802.11 de forma dinâmica são assuntos recorrentes, principalmente tratados utilizando algoritmos de atribuição de canal, os quais se subdividem em abordagens centralizadas e distribuídas. Nas abordagens centralizadas é necessário um controlador central de rede que consolida todas as informações coletas da rede e realiza as atribuições de canal. Os trabalhos [P. Mahonen 2004],[Mishra et al. 2006] e [Balbi et al. 2012] são bons exemplos de soluções centralizadas.

Nas abordagens distribuídas, cada ponto de acesso realiza sua atribuição de canal de forma autônoma com base em suas informações coletadas e dos PAs vizinhos. Por exemplo, os trabalhos de [Silva and Rezende 2006], [Gramacho et al. 2013], [Monteiro et al. 2016] e [Maturi et al. 2017] decidem e aplicam alguma política de alocação de canal de modo distribuído, ou seja, sem uma autoridade central.

Contudo, todas as abordagens necessitam realizar uma varredura do meio, seja ela de forma centralizada ou distribuída, tornando necessário o uso de soluções baseadas em aquisições de dados ou leituras do espectro de redes IEEE 802.11 na faixa de 2,4 GHz. Por exemplo, [Lee et al. 2013] e [Seng et al. 2012] propõem estudos sobre os efeitos das modulações nos dispositivos de redes sem fio, varrendo e analisando o espectro por meio da ferramenta inSSIDer. A exemplo do inSSIDer, a maioria das ferramentas de análise de espectro fornece somente relatórios sobre a ocupação do mesmo, cabendo ao usuário interpretar qual o melhor canal a ser utilizado. Diferente dessas ferramentas, o TriploS fornece a análise e informa explicitamente o melhor canal a ser utilizado.

3. TriploS

Como dito anteriormente, o *Smart Spectrum Sensing* (TriploS) é uma ferramenta criada para auxiliar na escolha do melhor canal de operação para redes IEEE 802.11 operando na faixa de frequência de 2.4 Ghz. A sugestão de melhor canal é feita através de cálculos de sobreposição realizados por meio de métricas coletadas, a partir da leitura dos canais em uso. A Figura 1 exemplifica esse processo através de um cenário com redes IEEE 802.11 preexistentes, onde o TriploS foi utilizado para sugerir o melhor canal de alocação para uma nova rede.

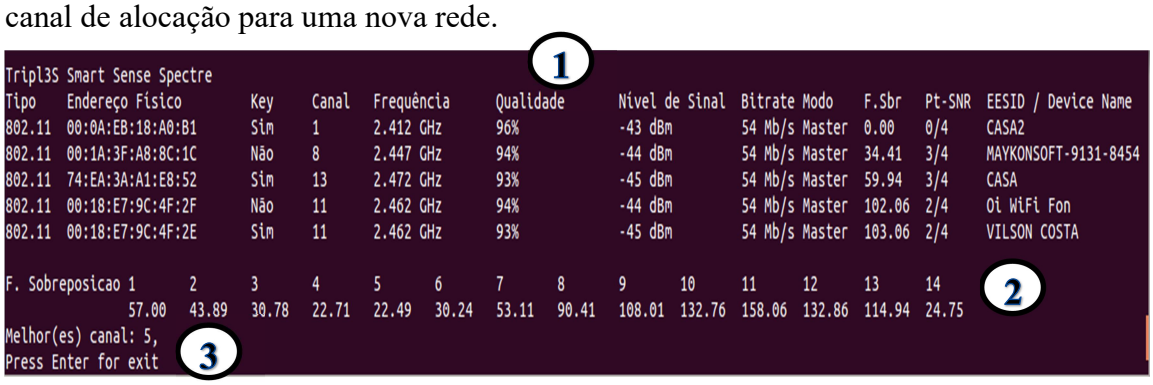


Figura 1 - 1) Métricas coletadas; 2) Cálculo do Fator de Sobreposição; 3) Melhor Canal definido.

Como pode ser visto na Figura 1, a ferramenta identificou cinco redes com características distintas observando as métricas como qualidade, nível de sinal, canal, entre outras (Figura 1, item 1). A ferramenta realiza então o cálculo de fator de

sobreposição, a partir das características obtidas dos canais em uso detectados, e informa os índices de sobreposição por cada canal pertencente ao padrão IEEE 802.11 (**Figura** 1, item 2). Por fim, mediante as informações capturadas e o cálculo de fator de sobreposição aplicado é definido um melhor canal para alocar o novo ponto de acesso (**Figura** 1, item 3), ou seja, um canal que reduzirá a interferência a partir de propagações em canais adjacentes. Vale ressaltar que o TriploS considera todos os canais do espectro IEEE 802.11, sobrepostos e não sobrepostos.

3.1. Arquitetura

A Figura 2 apresenta a arquitetura da ferramenta TriploS incluindo o fluxo principal das atividades e seus componentes principais (responsáveis pelos passos 1-3 da Figura 1). Os componentes são descritos em detalhes a seguir.

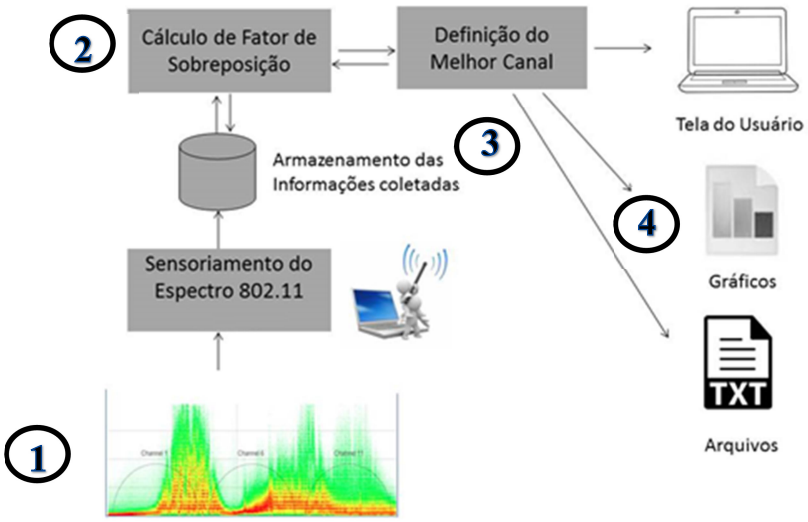


Figura 2 - Arquitetura do TriploS.

3.1.1. Sensoriamento do Espectro

O primeiro passo da ferramenta TriploS é a captura de quadros *beacon* recebidos por uma interface de rede IEEE 802.11. O objetivo dessa etapa é realizar um levantamento da utilização atual do espectro na faixa 2.4 GHz. O resultado desse processo corresponde à identificação das redes em operação, bem como um conjunto de métricas associadas a cada rede como endereço físico, tipo de encriptação da rede, canal, frequência, qualidade do link, nível de sinal, taxa de bits e modo de operação (ver Figura 1, item 1). Essas informações são então armazenadas para serem utilizadas pelo módulo de cálculo do fator de sobreposição.

Vale salientar que a ferramenta foi projetada para ser compatível com placas de rede sem fio comumente utilizadas em dispositivos móveis como *notebooks*, *tablets* e *smartphones*, constituindo um fator de fácil adoção. Atualmente a limitação está na necessidade da ferramenta operar sobre um sistema operacional baseado no kernel Linux.

3.1.2. Cálculo do Fator de Sobreposição

O segundo passo da ferramenta utiliza as informações coletadas durante o sensoriamento do espectro para então efetuar o cálculo do fator de sobreposição. No TriploS esse fator é computado baseado nos cálculos de interferência e sinal ruído

definidos por [Monteiro et al. 2016], que utiliza uma matriz gerada através da relação de sobreposição entre os canais adjacentes, ou seja, a porcentagem de intersecção entre os canais. Desta forma, a partir dos canais detectados em propagação são calculados os fatores de sobreposição (ver Figura 1, item 2).

3.1.3. Definição do Melhor Canal

O terceiro passo da ferramenta é a definição do melhor canal a ser utilizado. Neste trabalho, a sugestão é feita atualmente considerando o menor índice de sobreposição encontrado no passo anterior. Observando o exemplo dado na Figura 1, o melhor canal a ser utilizado seria o canal 5 (fator de sobreposição igual a 22.49).

3.1.4 Saída

A ferramenta TriploS exibe na tela do usuário todas as informações coletadas do espectro, o cálculo do fator de sobreposição e a definição do melhor canal, podendo ser gerados gráficos e arquivos contendo essas informações, a partir de intervalos de tempo definidos. Tais arquivos são utilizados por geradores de gráficos como o Gnuplot e ferramentas de análise comportamental da rede como o Iperf.

4. Detalhes de implementação da ferramenta TriploS

A ferramenta TriploS é totalmente baseada em tecnologias livres constituído por:

1. Uma ferramenta livre de coleta de informações do espectro de redes sem fio IEEE 802.11 na faixa de 2.4 GHz.
2. Armazenamento de informações coletadas em formato de arquivo, para visualização do comportamento do espectro a partir de ferramentas livres de visualização gráfica, a exemplo da ferramenta Gnuplot.
3. Avaliação e definição de melhor canal para uso, considerando todos os canais do espectro de redes sem fio IEEE 802.11.

Para desenvolvimento da ferramenta livre de coleta de informações foi utilizada a biblioteca denominada “iwlib.h” disponibilizada em linguagem de programação C para plataforma Linux. Esta biblioteca permite que informações do espectro possam ser coletadas, além de outras informações sobre a interface de rede sem fio, como ESSID (identificação da rede sem fio), nível de sinal, canal, frequência, qualidade e modo de operação.

Na execução da ferramenta TriploS é possível realizar o armazenamento das informações coletadas, a partir de instantes de tempo e intervalos de amostras definidos, conforme comando: “*triplos –n a –t b* ”, onde *a* é o número de amostras e *b* é o intervalo de tempo entre as amostras em segundos. Estes dados são armazenados individualmente para cada rede detectada, em um arquivo no formato texto (.txt), a partir da utilização do parâmetro “–f (file)”. Para visualizar os dados coletados em gráficos é necessária à adição do parâmetro “–c (charts)” na execução da ferramenta.

5. Avaliações e Resultados.

A avaliação da ferramenta foi realizada no Laboratório de Segurança de Sistemas e Tecnologias Emergentes (ETSS) na Universidade Federal do Amazonas. Alguns dos gráficos gerados são exemplificados na Figura 3 (índices de sobreposição de canais do espectro) e Figura 4 (ocupação do espectro por redes IEEE 802.11).

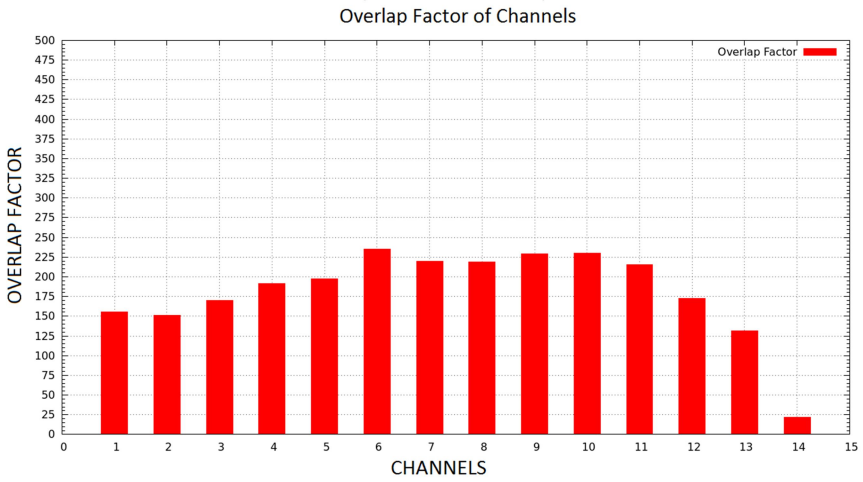


Figura 3 - Fator sobreposição calculado, a partir do espectro detectado.

A Figura 3 apresenta os índices de sobreposição obtidos pela execução da ferramenta, os quais são calculados pelo TriploS a partir dos dados coletados de ocupação do espectro, apresentados na Figura 4. O melhor canal é escolhido com base no menor índice de sobreposição encontrado.

A métrica vazão de rede foi escolhida para avaliar o impacto na interface de rede com o uso do TriploS. A vazão da rede foi obtida por meio da combinação da ferramenta TriploS com uma ferramenta de uso livre para medições e controle de rede *Iperf*, utilizada para mensurar a vazão gerando um tráfego UDP de 1470 bytes constante durante o intervalo de 600 segundos (10 minutos) para os seguintes intervalos de análise de tempo (0, 1, 5, 10, 15, 30, 45 e 60) entre as amostras.

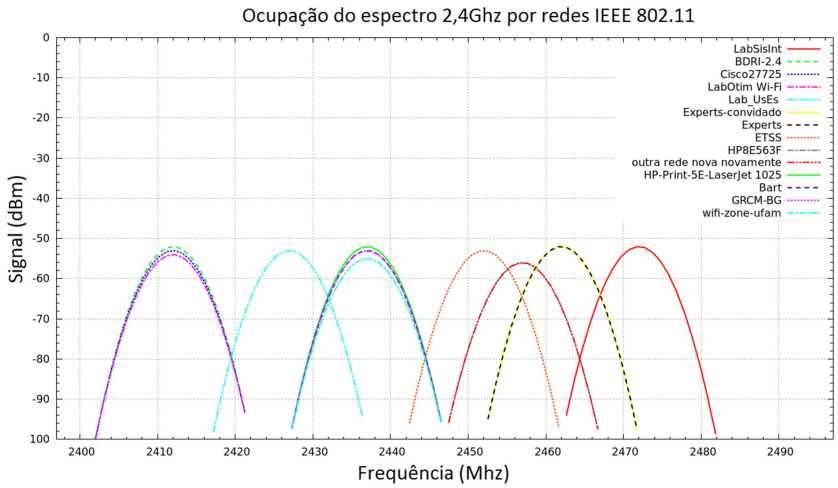


Figura 4 - Ocupação do espectro por redes IEEE 802.11.

A Figura 5 mostra que o impacto do TriploS sobre o desempenho da interface de rede utilizada diminui, na medida em que o intervalo de análise espectral é menor. Portanto, sugere-se que seja observado com muita cautela o intervalo de análise, visto que este tem influência direta no desempenho da interface de rede. Outra sugestão é fazer uso de duas interfaces de rede distintas, uma para a análise do tráfego de dados e outra exclusiva para sensoriamento do espectro.

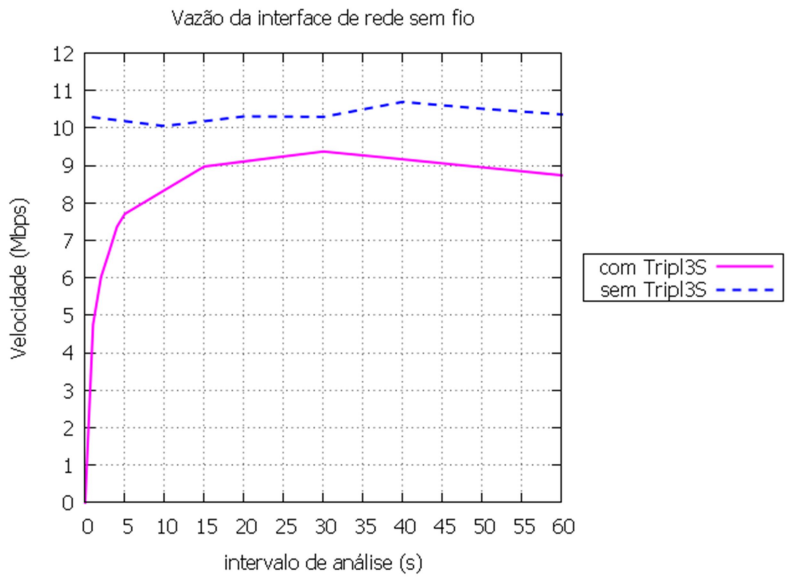


Figura 5 - Relação entre a vazão e o intervalo das amostras.

6. Demonstração

Para demonstrar o funcionamento do TriploS será montado um ambiente com um *notebook*, dois roteadores sem fio (RA e RB) e dois nós clientes (NA e NB). O *notebook* será utilizado na execução da ferramenta TriploS para identificar o melhor e pior canal, de acordo com o impacto gerado pela existência de outras redes sem fio padrão IEEE 802.11 sobre os roteadores RA e RB, no ambiente de demonstração. A partir das informações obtidas do espectro, o roteador RA será configurado com o pior canal (o que possui mais redes compartilhando sua frequência ou em frequências adjacentes), enquanto que RB será configurado com o melhor canal (menor quantidade de redes propagando em sua frequência ou em frequências adjacentes), apresentados pelo TriploS. Simultaneamente, serão estabelecidas conexões entre os roteadores e os nós clientes. A ferramenta Iperf será utilizada para demonstrar a melhoria da vazão quando o usuário configura o canal do roteador sem fio seguindo a informações fornecidas pelo TriploS.

7. Considerações Finais

Este trabalho apresentou o TriploS (*Smart Spectrum Sensing*), uma ferramenta que visa contribuir para análise de canais IEEE 802.11 para pontos de acesso sem fio. A partir da análise de dados coletados, o TriploS é capaz de sinalizar o canal mais apropriado considerando as características de sobreposição de canais do espectro, em que um determinando PA será inserido. Os resultados gerados pela ferramenta são usados como entrada para outras ferramentas de geração de gráficos (e.g Gnuplot) e de análise comportamental da rede (e.g. Iperf), conforme mostrado aplicação na seção 5. O TriploS pode ser usado para diminuir o esforço envolvido no desenvolvimento de mecanismos, algoritmos e/ou esquemas de seleção dinâmica de canal que utilizem cenários reais de testes. Como trabalhos futuros, almeja-se portar esta ferramenta para plataformas móveis como smartphones e tablets, a fim de maximizar e flexibilizar a análise e controle de espectro de dispositivos que utilizem a banda ISM, além de incluir outras características do espectro, tais como: outros padrões IEEE que utilizem a Banda ISM 2.4 GHz (Bluetooth ou telefones sem fio) e efeitos de multipath fading de paredes, o que tornará a definição de melhor canal ainda mais confiável.

A ferramenta TriploS e seu manual de uso e de instalação estão disponíveis para download em: http://etss.ufam.edu.br/index.php/ferramentas/66-triplos e em https://github.com/engalexmonteiro/triplos.

Agradecimentos

Este trabalho foi parcialmente financiado pela Comissão Europeia e CNPQ através do projeto de pesquisa IMPRESS (FP7-ICT-2013-EU-Brazil, GA No. 614100), pela FAPEAM através do Projeto de Pesquisa PROTI Amazônia.

Referências

Balbi, H., Souza, F. R., CARRANO, R. C., et al. (2012). Algoritmo de seleção de canais centralizado para redes IEEE 802 . 11 com controlador. *Global Information Infrastructure and Networking Symposium (GIIS)*, v. V.1, p. 1–7.

Bhartia, A., Chakrabarty, D., Chintalapudi, K., et al. (2016). IQ-Hopping : Distributed Oblivious Channel Selection for Wireless Networks. *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, p. 81–90.

Dugan, J., Elliott, S., Mah, B. A., Poskanzer, J. e Prabhu, K. (2017). Iperf. https://iperf.fr/.

Farproc (2017). WiFi analyzer. http://a.farproc.com/wifi-analyzer.

Gnuplot (2017). Gnuplot. http://www.gnuplot.info/.

Gramacho, S., Araujo, M. e Figueiredo, G. (2013). Dinâmica de Seleção de Melhores Canais em Redes IEEE 802 .11 com Modelo de Interferência CCA / SINR. *XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, v. 1, p. 88–101.

IEEE Std 802.11, I. S., LAN/MAN, C. e Society, C. I. (2012). *IEEE Standard for Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*. v. 2007

Maturi, F., Gringolit, F. e Cigno, R. Lo (2017). A Dynamic and Autonomous Channel Selection Strategy for Interference Avoidance in 802 . 11. p. 1–8.

Metageek (2017). inSSIDer 4. http://www.metageek.com/products/inssider/personal/.

Mishra, A., Shrivastava, V., Agrawal, D., Banerjee, S. e Ganguly, S. (2006). Distributed Channel Management in Uncoordinated. *Proceedings of the 12th annual international conference on Mobile computing and networking*, p. 170–181.

Monteiro, A., Souto, E., Pazzi, R. e Kiljander, J. (2016). Atribuição dinâmica de canais em redes sem fio não coordenadas IEEE 802 . 11 , baseada em fatores de sobreposição e intensidade de sinal. *XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, p. 704–717.

P. Mahonen, J. R. e M. P. (2004). Automatic channel allocation for small wireless local area networks using graph colouring algorithm approach. *15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, v. 1, p. 536–539.

Silva, M. R. da S. e Rezende, J. F. (2006). A Dynamic Channel Allocation Mechanism for IEEE 802 .11 Networks. p. 403–408.

Realização

Apoio Fomento

Apoio Institucional

Patrocinador Diamante

Núcleo de Informação e Coordenação do Ponto BR

Comitê Gestor da Internet no Brasil

Patrocinador Ouro

Patrocinador Bronze