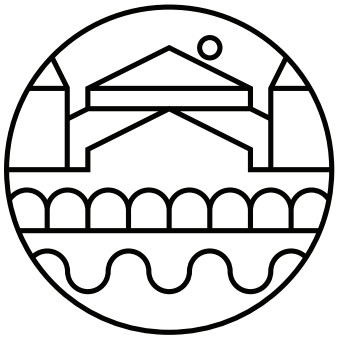
XXXV
SIMPÓSIO BRASILEIRO DE
REDES DE COMPUTADORES
E SISTEMAS DISTRIBUÍDOS

15 a 19 de maio de 2017
Belém - Pará

Anais VII WoSiDA 2017



XXXV
SIMPÓSIO BRASILEIRO DE
REDES DE COMPUTADORES
E SISTEMAS DISTRIBUÍDOS

15 a 19 de maio de 2017
Belém - Pará

Anais do VII WoSiDA 2017
Workshop on Autonomic Distributed System

Editora
Sociedade Brasileira de Computação (SBC)

Organização
Allan Edgard Silva Freitas (IFBA)
Emanuel Ferreira Coutinho (UFC)
Ronaldo Alves Ferreira (UFMS)
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

VII Workshop on Autonomic Distributed System (7: 2017: Belém, Pa).

Anais / VII Workshop on Autonomic Distributed System – WoSiDA; organizado por Antônio Jorge Gomes Abelém, Eduardo Coelho Cerqueira, Ronaldo Alves Ferreira, Allan Edgard Silva Freitas, Emanuel Ferreira Coutinho - Porto Alegre: SBC, 2017

32 p. il. 21 cm.
Vários autores
Inclui bibliografias

1. Redes de Computadores. 2. Sistemas Distribuídos. I. Abelém, Antônio Jorge Gomes II. Cerqueira, Eduardo Coelho III. Ferreira, Ronaldo Alves IV. Freitas, Allan Edgard Silva V. Coutinho, Emanuel Ferreira 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 do Coordenador de Workshops

É com grande prazer que os convido a prestigiar os workshops do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) nos dias 15, 16 e 19 de maio de 2017. Tradicionalmente, os workshops abrem e fecham a semana do SBRC e são responsáveis por atrair uma parcela expressiva de participantes para o Simpósio. Como coordenador de workshops, dividi com os coordenadores gerais do SBRC a nobre tarefa de selecionar os workshops que melhor representam a comunidade e que fortaleçam novas linhas de pesquisa ou mantenham em evidência linhas de pesquisa tradicionais.

Em resposta à chamada aberta de workshops, recebemos dez propostas de alta qualidade, das quais nove foram selecionadas. Além disso, mantivemos a longa colaboração com a RNP por meio da organização do WRNP, que já é uma tradição na segunda e terça-feira da semana do SBRC. Dentre as propostas aceitas, sete são reedições de workshops tradicionais do SBRC que já são considerados parte do circuito nacional de divulgação científica nas várias subáreas de Redes de Computadores e Sistemas Distribuídos, como o WGRS (Workshop de Gerência e Operação de Redes e Serviços), o WTF (Workshop de Testes e Tolerância a Falhas), o WCGA (Workshop em Clouds, Grids e Aplicações), o WP2P+ (Workshop de Redes P2P, Dinâmicas, Sociais e Orientadas a Conteúdo), o WPEIF (Workshop de Pesquisa Experimental da Internet do Futuro), o WoSiDA (Workshop de Sistemas Distribuídos Autonômicos) e o WoCCES (Workshop de Comunicação de Sistemas Embarcados Críticos). Como novidade, teremos dois novos workshops com programação diversificada e grande apelo social, o CoUrb (Workshop de Computação Urbana) e o WTICp/D (Workshop de TIC para Desenvolvimento).

Temos certeza que 2017 será mais um ano de sucesso para os workshops do SBRC pelo importante papel de agregação que eles exercem na comunidade científica de Redes de Computadores e Sistemas Distribuídos no Brasil.

Aproveitamos para agradecer o apoio recebido de diversos membros da comunidade e, em particular, a cada coordenador de workshop, pelo brilhante trabalho. Como coordenador dos workshops, agradeço imensamente o apoio recebido da Organização Geral do SBRC 2017.

Esperamos que vocês aproveitem não somente os workshops, mas também todo o SBRC e as inúmeras atrações de Belém.

Ronaldo Alves Ferreira

Coordenador de Workshops do SBRC 2017

Mensagem dos Coordenadores do VII WoSiDA 2017

Atualmente sistemas computacionais têm sido caracterizados pela presença de requisitos cada vez mais rigorosos em relação a escalabilidade, disponibilidade, dependabilidade, suporte a heterogeneidade, eficiência energética, facilidade de integração e extensibilidade. Adicionalmente, um dos principais desafios de pesquisa na atualidade é garantir que um sistema continue apresentando uma qualidade de serviço adequada e aceitável, mesmo operando sob ambientes dinâmicos e incertos.

Nesse contexto, a tomada de decisão em tempo de projeto, baseada em suposições sobre os dados envolvidos e gerados, características da carga de trabalho aplicada, e da plataforma de computação e comunicação, pode não ser mais efetiva devido à grande variabilidade que tais fatores apresentam ao longo da operação do sistema.

Uma abordagem que se mostra promissora é a transferência deste processo de tomada de decisão para o tempo de execução, apoiada por uma infraestrutura de monitoramento do ambiente e do próprio sistema, pela análise dos dados coletados, pelo planejamento de adaptações e pela realização destas adaptações no sistema em execução. E esta a abordagem central presente em iniciativas tais como a computação autonômica, os sistemas auto-adaptativos e os sistemas auto-gerenciados. Ao longo dos anos, os resultados avançaram de simples algoritmos adaptativos e arquiteturas reconfiguráveis para, por exemplo, verificação e validação em tempo de execução, tratamento de incertezas, adaptações descentralizadas em sistemas distribuídos de larga escala e aplicações em áreas tais como sistemas cyber-físicos, computação em nuvem e internet das coisas.

Diversos aspectos computacionais podem ser alvo de tais adaptações, o que faz com que o assunto vigore nas pautas de pesquisa de áreas tais como robótica, gerência de redes, engenharia de software, computação bio-inspirada, inteligência artificial, sistemas distribuídos e sistemas tolerantes a falhas, dentre outras.

O Workshop de Sistemas Distribuídos Autonômicos (WoSiDA) tem como objetivo viabilizar a troca de experiências e interesses de pesquisa relacionados à computação autonômica, com foco na sua utilização como infraestrutura básica para construção de sistemas distribuídos modernos. O WoSiDA acomoda não somente discussões relacionadas a modelos, algoritmos e formalismos para sistemas distribuídos autonômicos, mas também debate aspectos associados a plataformas, ferramentas, metodologias e experimentações de areas tais como engenharia de software, sistemas cyber-físicos e sistemas de automação e controle.

Um dos principais objetivos do WoSiDA é catalisar discussões relacionadas à problemática decorrente da demanda conjunta por distribuição e auto-gerenciamento. Dentre os principais tópicos de interesse do WoSiDA, destacam-se: modelos e algoritmos para sistemas distribuídos autonômicos, aspectos de projeto de sistemas distribuídos autonômicos, modelagem e análise de sistemas distribuídos autonômicos, middleware e programação de sistemas distribuídos autonômicos, verificação e validação, dependabilidade em sistemas distribuídos autonômicos, técnicas para a auto-organização, abordagens bio-inspiradas para sistemas distribuídos autonômicos e redes autonômicas.

Nesta sétima edição do workshop, foram selecionados cinco artigos abordando as temáticas acima.

Acreditamos que os artigos selecionados proporcionarão ótimas discussões discussões, atendendo aos objetivos do workshop, atuando como agentes motivadores para o fortalecimento dos grupos de pesquisa já existentes e formação de novas colaborações.

Allan Edgard Silva Freitas e Emanuel Ferreira Coutinho

Chairs do WoSiDA 2017

Comitê de Programa

- Alirio Sá (UFBA, Brazil)
- Allan Freitas (IFBA, Brazil)
- Alysson Bessani (University of Lisbon, Portugal)
- Antonio Casimiro (University of Lisbon, Portugal)
- Diego Garcia (UFOP, Brazil)
- Emanuel Coutinho (UFC, Brazil)
- Fabio Costa (UFG, Brazil)
- Flávio Assis Silva (UFBA, Brazil)
- Francisco Cruz (University of Minho, Portugal)
- Luis Rodrigues (INESC-ID/IST, Portugal)
- Marcos Barreto (UFBA, Brazil)
- Mauro Oliveira (IFCE, Brazil)
- Nazim Agoulmine (University of Evry, France)
- Raimundo Macedo (UFBA, Brazil)
- Sand Correa (UFG, Brazil)
- Sandro Andrade (IFBA, Brazil)
- Sérgio Gorender (UFBA, Brazil)
- Tales Heimfarth (UFLA, Brazil)

Sumário

VII Workshop On Autonomic Distributed Systems (WoSiDa)

SBRC 2017

Sessão Técnica 1

Uma Experiência de Adaptação de Quóruns para Detecção de Falhas em Consenso Byzantino no Modelo Síncrono Particionado

Wellington L. S. da Silva[1], **Marco Antônio D. Ramos**[1], **Raimundo José de A. Macêdo**[1]

[1]Laboratório de Sistemas Distribuídos (LaSiD)
Departamento de Ciência da Computação
Instituto de Matemática
Universidade Federal da Bahia
Campus de Ondina – Salvador, BA – Brasil

wellingtonlss@ufba.br, madr@uesb.edu.br, macedo@ufba.br

Abstract. *The development of byzantine-fault-tolerant distributed systems presents considerable challenges given the arbitrary character of such faults. The coordinated use of multiple fault tolerance strategies together with exploiting the characteristics of the system at hand is key to system strengthening. This article presents preliminary results of the implementation of an adaptable quorum system for byzantine fault tolerance in partitioned synchronous systems which will allow to exploit the peculiar characteristics of this kind of system.*

Resumo. *O desenvolvimento de sistemas distribuídos tolerantes a falhas bizantinas apresenta desafios consideráveis, dado o caráter arbitrário desse tipo de falha. O uso coordenado de múltiplas estratégias de tolerância a falhas com o aproveitamento das características do sistema em questão torna-se chave para o robustecimento do sistema. Este artigo apresenta resultados preliminares da implementação de um sistema de quóruns adaptável para tolerância a falhas bizantinas em sistemas síncronos particionados que aproveita as características peculiares deste tipo de sistema.*

1. Introdução

Sistemas distribuídos são sistemas cujos componentes estão situados em computadores conectados por redes de comunicações e se coordenam e comunicam apenas através de trocas de mensagens. Tais componentes são chamados genericamente de "processos". Não existem relógios globais, todos os processos do sistema são considerados inerentemente concorrentes e falham de modo independente uns dos outros [COULORIS et al. 2011]. Limites temporais de processamento e entrega de mensagens permitem classificar os sistemas em síncronos, nos quais todos os limites são conhecidos, e assíncronos, em que todos os limites são inexistentes, além de outros modelos intermediários [DWORK et al. 1988]. Essas características tornam desafiadoras certas tarefas, como obter concordância dos processos em torno de algum valor, uma classe de problemas conhecida como os Problemas de Consenso, relevante para este trabalho.

Sistemas distribuídos tolerantes a falhas apresentam como característica principal a resiliência do sistema à ocorrência de uma ou mais falhas de processos, até um número limite f de processos faltosos, além do qual o sistema colapsa [COULORIS et al. 2011].

O tipo de falha a tolerar é uma restrição importante do sistema, pois diferentes tipos de falhas implicam na necessidade de diferentes de mecanismos de resiliência. As falhas bizantinas são falhas nas quais o processo que falha se desvia arbitrariamente de sua especificação. Lamport, Shostak e Pease [LAMPORT et al. 1982] demonstraram que as falhas bizantinas impõem limites f menores do que falhas mais simples. Um resultado importante na teoria de sistemas distribuídos, a "Impossibilidade FLP", [FISHER et al. 1985], mostra que a sincronia temporal também é uma restrição importante na elaboração de mecanismos de resiliência para sistemas distribuídos.

Duas das principais estratégias por trás de mecanismos de resiliência são a redundância de processos e a detecção de falhas. Uma forma de implementar redundância de processos é o estabelecimento de sistemas de quóruns. Vários algoritmos distribuídos se baseiam apenas em sistemas de quóruns para tolerar falhas, como, por exemplo, o algoritmo Paxos e sua variante Paxos Bizantino [LAMPORT 2010].

Detectores de defeitos não confiáveis, introduzidos por Chandra e Toueg [CHANDRA and TOUEG 1996], são módulos conectados aos processos e que atuam como oráculos, informando aos processos se outros processos falharam por parada ou não. A informação não necessita estar correta o tempo todo e o detector pode restabelecer o status de um processo que marcou como faltoso anteriormente. Diversos detectores de defeitos para falhas bizantinas foram introduzidos por diferentes autores após o trabalho de Chandra e Toueg. Um dos primeiros trabalhos na área foi o de Kihlstrom *et al* [KIHLSTROM et al. 2003], no qual mostram uma estratégia de resiliência que utiliza um sistema de quóruns e um detector de defeitos para falhas bizantinas que funcionam em complementaridade um com o outro.

O objetivo deste artigo é apresentar resultados preliminares acerca da implementação de um sistema de quórums adaptável para o mascaramento de falhas, que faz parte de uma estratégia mista de tolerância de falhas bizantinas. Diferentemente da estratégia de Kilhstrom *et al*, que consiste na utilização de um quórum fixo, o modelo de sistema utilizado neste trabalho comporta introduzir um sistema de quóruns bizantino adaptável ao invés do quórum fixo original, o que permite que os processos faltosos sejam efetivamente excluídos do sistema, melhorando sua robustez geral.

Os conteúdos estão organizados da seguinte forma: A seção 2 apresenta a revisão da literatura pertinente, a seção 3 apresenta o mecanismo de quóruns adaptáveis utilizado. A seção 4 apresenta trabalhos correlatos e a seção 5 apresenta os resultados preliminarmente obtidos e as conclusões deste trabalho.

2. Modelo de Sistema e Quóruns Adaptativos

2.1. Modelo Síncrono Particionado

O modelo fundamental do sistema utilizado neste trabalho é o modelo síncrono particionado [MACÊDO and GORENDER 2009]. Um **sistema no modelo síncrono-particionado** é composto de um conjunto $\Pi = \{p_1, p_2, .., p_n\}$ de processos possivelmente situados em computadores distintos e um conjunto $\mathcal{X} = \{c_1, c_2, .., c_m\}$ de canais de comunicação que conectam processos distintos dois-a-dois permitindo a comunicação bidirecional entre eles.

Os canais de comunicação são confiáveis, ou seja, não corrompem nem perdem

mensagens. Tanto processos quanto canais de comunicação podem ser *timely* ou *untimely*, de modo semelhante à definição de síncrono/assíncrono de [DWORK et al. 1988]. Os limites δ, máximo atraso na entrega de mensagens, e ϕ, máxima diferença de velocidade de processamento entre os processadores, são garantidos. Um módulo oráculo de *timeliness*, informa se processos e canais de comunicação são *timely* ou não. O sistema apresenta ainda a propriedade *sincronia particionada forte*, isto é, todo processo pertence a alguma partição síncrona. Essa propriedade garante que todos os processos possuam ao menos uma conexão *timely*.

2.2. Falhas Bizantinas

Falhas bizantinas são falhas nas quais a execução dos processos faltosos de um sistema desvia arbitrariamente de sua especificação original. Elas foram caracterizadas em [LAMPORT et al. 1982]. O artigo mostra que num sistema distribuído síncrono, na presença de processos bizantinos, o número de processos que falham f deve ser menor que um terço do número total de processos n, ou seja, $n \geq 3f + 1$. Dwork, Lynch e Stokemeyer [DWORK et al. 1988] mostraram que esta proporção vale para sistemas parcialmente síncronos. Isso torna a resiliência a processos bizantinos um processo de alto custo operacional.

Além da questão do custo, processos bizantinos podem não falhar de modo consistente. Kihlstrom *et al* [KIHLSTROM et al. 2003] classificam as falhas bizantinas em observáveis e não observáveis. As falhas observáveis podem ser por omissão, corrupção de mensagens ou mensagens mutantes. No primeiro caso o processo faltoso omite mensagens No segundo caso, o processo envia mensagens mal formadas ou injustificadas. No terceiro caso envia mensagens válidas mas diferentes para diferentes processos. As falhas bizantinas não observáveis não manifestam efeitos externos e podem apenas ser mascaradas.

2.3. Sistemas de quórum

Um **sistema de quórum** para uma certa coleção de processos num sistema distribuído é um conjunto de subconjuntos desses processos, onde cada par de subconjuntos se intersecta de modo significativo. Cada subconjunto do sistema é chamado **quórum**. Os quóruns podem variar de processo para processo, mas a intersecção desses subconjuntos deve conter processos corretos em número suficiente para garantir a tolerância às falhas previstas no modelo de falhas de processos do sistema [MALKHI and REITER 1997]. Assim, para tolerar f falhas de parada, a intersecção deve conter ao menos um processo correto, enquanto que para tolerar o mesmo número de falhas bizantinas, a intersecção deve conter ao menos $2f + 1$ processos corretos [LAMPORT et al. 1982]. Sistemas de quórum capazes de tolerar falhas bizantinas são chamados **quóruns bizantinos** [LAMPORT 2010].

Um sistema de quóruns é adaptável quando o número de processos no quórum varia ao decorrer da execução do sistema. Os parâmetros do sistema de quóruns são recalculados para garantir a resiliência do sistema. Como cada processo do sistema distribuído vai calcular um certo quórum num dado momento, os quóruns só vão finalmente concordar após um tempo, isto é, excluir do quórum todos os processos considerados bizantinos.

3. Adaptação de Quóruns para o Consenso Bizantino

O algoritmo de Kihlstrom *et al* para o consenso é dividido em 5 tarefas que controlam o fluxo de mensagens do protocolo de consenso e a saída do detector. Cada fase das tarefas 1, 2 e 3 estabelece um quórum necessário para o processo progredir corretamente, sempre baseado no número total de processos do sistema. Apenas a fase 3 da tarefa 1 controla se o líder está na saída do detector. A estratégia consiste no compartilhamento de responsabilidades entre o sistema de quóruns e o detector de defeitos. O detector garante a propriedade de *liveness* do sistema por dois mecanismos: através do monitoramento o líder quando este envia a mensagem *SELECT* sozinho no passo 2 do protocolo e através do envio de ecos de mensagens, isto é, reenvio das mensagens corretas recebidas para todos os outros processos. Os processos podem identificar variações inadmissíveis nas mensagens ao receber o eco. O eco também possibilita a recepção indireta de uma mensagem atrasada. O sistema de quóruns garante *safety* ao permitir apenas decisões válidas segundo os limites do protocolo e mascara as falhas bizantinas não detectáveis.

No contexto do modelo síncrono particionado as condições são diferentes. Processos que compartilham uma partição síncrona conhecem todos os limites temporais dos outros processos da partição. Isso implica que um processo que atrasa não pode estar se comportando dentro da especificação. Resta definir a partir de quando um processo p_i pode estabelecer um *timeout* estrito para um outro processo p_j de sua partição, e nesse ponto as mensagens de eco desempenham um papel que não possuem no sistema de Kihlstrom *et al*.

O processo p_i, ao ecoar uma mensagem para o processo p_j dentro de sua própria partição síncrona pode estabelecer um *timeout* estrito para que p_j envie a mensagem resposta esperada pelo protocolo. Dentro da mesma partição, todos os processos e canais de comunicação são *timely*, portanto os limites superiores δ e ϕ se aplicam e são conhecidos, o que implica que processos que infringem estes limites estão operando fora de sua especificação e podem ser considerados bizantinos. Essa é uma condição que se verifica em sistemas síncronos particionados e não se verifica em sistemas parcialmente síncronos baseados em GST.

Ainda em relação ao algoritmo de Kihlstrom *et al*, a tarefa 5, que controla a saída do detector de defeitos, também ganhou novo papel. A cada recebimento de notificação de detecção, caso um número adequado de notificações tenha sido recebido dos processos da mesma partição de algum processo p_j a respeito de p_j, o processo é retirado do quórum e novos parâmetros de resiliência são calculados automaticamente, assim como todos os quóruns das outras tarefas. Caso o processo suspeito seja o líder, então uma nova rodada é iniciada. As decisões anteriores só são revogadas neste último caso. Se o líder é correto, então as decisões são consideradas corretas por conta da correção *as is* de algum processo que venha a falhar posteriormente.

4. Trabalhos Correlatos

Haeberlen, Kousnetsov e Druschel [HAEBERLEN et al. 2006] apresentam o sistema PeerReview, que utiliza um detector de falhas bizantinas também inspirado em Kihlstrom. Os autores, porém, utilizam o detector não para controlar os processos a partir de atrasos no envio de mensagens e verificação de mensagens, mas através de um histórico não forjável de envios e recebimentos das mesmas. O algoritmo de consenso mascara apenas as

falhas não-detectáveis.

Duan *et al* [DUAN et al. 2014] utilizam tolerância a falhas bizantinas integrada a um sistema de detecção de intrusões para robustecer uma máquina replicada de estados (*replicated state machine*) que provê resiliência para falhas bizantinas. Os autores também procuram contornar o limite de $3f + 1$ em ambiente assíncrono em múltiplos domínios.

Bousbiba e Echtle [BOUSBIBA and ECHTLE 2016] apresentam um protocolo de diagnóstico e concordância bizantina, que detecta a ocorrência de falhas bizantinas e ao mesmo tempo provoca um diagnóstico consensual acerca do processo faltoso. Os autores também distribuem responsabilidades de tolerância a falhas entre detector e protocolo de consenso, mas no caso para redes síncronas.

Diferente destes trabalhos, nossa proposta se baseia numa solução de compromisso entre detecção de defeitos e sistema de quóruns adaptativo, que aproveita as propriedades do sistema síncrono particionado para oferecer tolerância a falhas bizantinas ao modelo com a possibilidade de utilizar quóruns menos restritivos que $3f + 1$.

5. Resultados e Conclusões

Esta seção retrata um exemplo do conjunto de simulações realizadas até o momento para constatar o funcionamento dos mecanismos anteriormente descritos. Foram realizadas até o momento 760 de um total de 1200 simulações em diversas configurações de sistema para 3 cenários principais: Kihlstrom sobre GST (GST), Kilhstrom sobre síncrono particionado (MIX) e o Kihlstrom adaptativo sobre síncrono particionado (SPA).

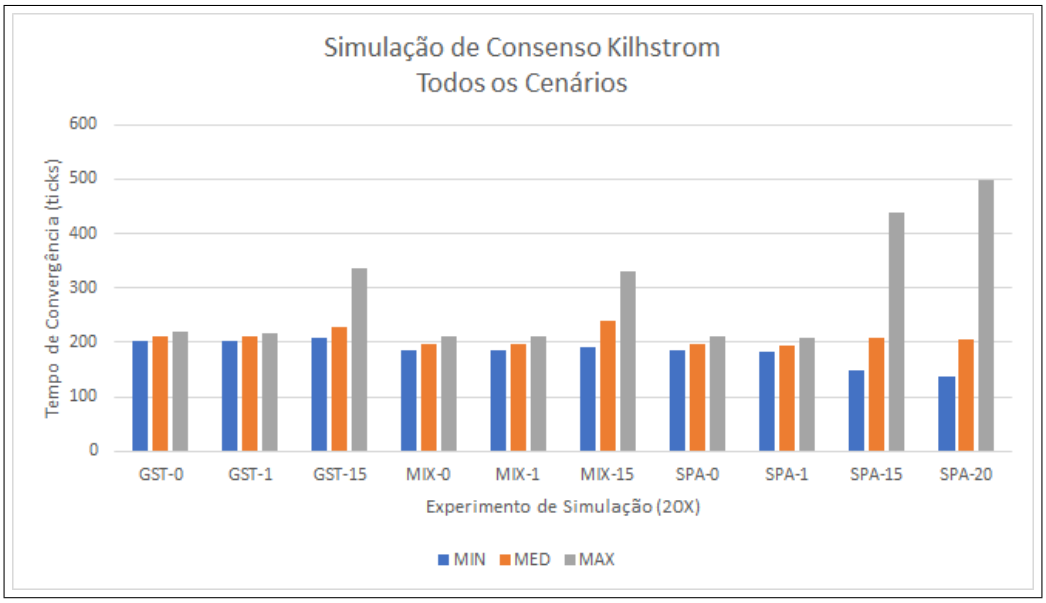


Figure 1. Detector de Kihlstrom em todos os cenários. O cenário SPA-20, com 20 falhas, demonstra a adaptação de quórum discutida neste trabalho.

O gráfico na (Figura 1) apresenta um *snapshot* dos dados com os tempos médios de convergência do algoritmo de consenso de Kihlstrom *et al* para 50 processos, em todos os cenários testados, com 20 simulações por cenário. Os cenários MIX e SPA utilizaram 5 partições com 10 processos cada. Foram injetadas respectivamente 0, 1, 15 e, no caso do

SPA, com 20 falhas bizantinas. O cenário SPA-20 demonstra o quórum adaptativo com 20 detecções e falha por corrupção de mensagens. O resultado obtido até o momento mostra que o quórum adaptativo permite superar o limite de Lamport para $f \leq \lfloor (n-1)/3 \rfloor = 15$ processos. Estabeleceu-se um limite para 20 processos como prova de conceito. Os próximos trabalhos se concentrarão na consolidação dos resultados obtidos e demonstração da resiliência mínima do sistema.

6. Referências

BOUSBIBA, O. and ECHTLE, K. (2016). A fast byzantine fault-tolerant diagnostic agreement protocol for synchronous distributed systems. In *ARCS 2016; 29th International Conference on Architecture of Computing Systems*, pages 1–11.

CHANDRA, T. D. and TOUEG, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267.

COULORIS, G., DOLLIMORE, J., KINDBERG, T., and BLAIR, G. (2011). *Distributed Systems: Concepts and Design*. Addison-Wesley Publishing Company, USA, 5 edition.

DUAN, S., LEVITT, K., MELING, H., PEISERT, S., and ZHANG, H. (2014). Byzid: Byzantine fault tolerance from intrusion detection. In *2014 IEEE 33rd International Symposium on Reliable Distributed Systems*, pages 253–264.

DWORK, C., LYNCH, N., and STOCKMEYER, L. (1988). Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323.

FISHER, M., LYNCH, N., and PATTERSON, M. (1985). Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382.

HAEBERLEN, A., KOUZNETSOV, P., and DRUSCHEL, P. (2006). The case for byzantine fault detection. In *Proc. of the 2nd Conference on Hot Topics in System Dependability. USENIX.*, volume 2, United States.

KIHLSTROM, K. P., MOSER, L. E., and MELLIAR-SMITH, P. M. (2003). Byzantine fault detectors for solving consensus. *Comput. J.*, 46(1):16–35.

LAMPORT, L. (2010). Byzantizing paxos by refinement. Technical report, Microsoft.

LAMPORT, L., PEASE, M., and SHOSTAK, R. (1982). The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401.

MACÊDO, R. J. and GORENDER, S. (2009). Perfect failure detection in the partitioned synchronous distributed system model. In *Availability, Reliability and Security, 2009. ARES'09. International Conference on*, pages 273–280. IEEE.

MALKHI, D. and REITER, M. (1997). Byzantine quorum systems. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 569–578, New York, NY, USA. ACM.

Autonomic Byzantine Replication for Intrusion Tolerance in Cyber-Physical Systems

Raimundo José de Araújo Macêdo

Laboratório de Sistemas Distribuídos (LaSiD)
Departamento de Ciência da Computação, Universidade Federal da Bahia
Campus de Ondina, CEP: 40170-110, Salvador-BA, Brasil
e
Universidade Federal do Sul da Bahia
Núcleo Protic/Salvador. Av. 7 de Setembro 2209, CEP: 40080-002, Salvador-BA, Brasil

macedo@ufba.br

Abstract. *Safety-critical applications of Internet-of-Things (IoT) enabled Cyber-Physical Systems (CPS) must cope with a large number of heterogeneous devices and systems with distinct computing and communication capabilities. Such physical and cyber interactions may result in system behaviors which cannot always be precisely anticipated at system design. Assuring dependable, secure, and timely system operations during runtime in these scenarios is a design challenge. In particular, when security attacks in such complex CPS result in the complete control of a system component, detecting and recovering from these faults is hard to handle since the compromised component can behave in an arbitrary manner. In this position paper, a short state-of-art overview of byzantine replication for implementing intrusion tolerance is presented and we discuss and argue for the usage of autonomic byzantine replication to enhance intrusion tolerance in such scenarios.*

Resumo. *Aplicações críticas em sistemas ciberfísicos baseados na Internet-das-Coisas têm que lidar com grande quantidade de dispositivos e sistemas heterogêneos, com enorme variedade em suas capacidades de processamento e comunicação. Tais interações entre o mundo físico e o mundo computacional, pode resultar em comportamentos que são difíceis ou impossíveis de serem antecipados em tempo de projeto. Nesses cenários, é desafiador para o projetista assegurar operações confiáveis ("dependable"), seguras e tempestivas. Em particular, quando ataques à segurança resultam no controle completo de componentes do sistema, detectar e ser recuperar de tais invasões é muito difícil devido ao comportamento imprevisível do invasor. Neste artigo (position paper), incialmente é apresentada uma revisão do estado-da-arte da literatura sobre tolerância à intrusão baseada em replicação bizantina, para, em seguida, ser proposto e discutido o uso de* ***replicação bizantina autonômica*** *para lidar e melhorar a capacidade de tolerar intrusões nesses cenários.*

1. Introdução

Integration of cyber and physical components has led to the development of new critical complex systems called Cyber-Physical Systems (CPS) [11], and the related paradigm of Internet of Things (IoT) is involved in the deployment of these CPS [14]. As such, many safety-critical applications of IoT-enabled CPS must cope with a large number of heterogeneous devices and systems with distinct computing and communication

capabilities. Such physical and cyber interactions may result in system behaviors which cannot always be precisely anticipated at system design. Assuring dependable, secure, and timely system operations during runtime in these scenarios is a design challenge. In particular, when security attacks in such complex CPS result in the complete control of a system component, detecting and recovering from these faults is hard to handle since the compromised component can behave in an arbitrary manner.

Researchers have advocated the use of intrusion tolerance mechanisms that allow a system to keep working properly even when intruders control some of its components [1, 2, 12]. The malicious attacks are modelled as arbitrary or byzantine faults [3], and many byzantine fault tolerance mechanisms have been proposed [4,5]. Due to the high complexity and dynamic nature of IoT-enabled CPS, autonomic solutions are required, so further research is needed to study component interactions and to develop effective automatic and self-adaptive intrusion tolerant mechanisms. This will allow for autonomously recovering the system functionality in a timely fashion, or degrading the system to a safe operation [6,9,10,13]. We have successfully applied such autonomic or self-adaptive mechanisms to implement several fault tolerant distributed protocols, such as failure detection and group communication [7,8]. Though intrusion tolerance allows the system under attack to work properly, prevention of intrusions is needed to avoid further failures when system resources become depleted. Intrusion tolerance should be complemented with intrusion prevention and intrusion detection and they should work in a coordinated and integrated fashion [2,12].

In this position paper we discuss and argue for the usage of autonomic byzantine replication to enhance intrusion tolerance. The presented approach combines our previous results in adaptive byzantine replication [5] and autonomic mechanisms for crash-resilient fault tolerance [7,8].

2. Replicated State Machines for Intrusion Tolerance

Attacks to IoT-enabled CPS should be prevented or detected so that system properties are preserved at all times. However, attacks cannot always be completed prevented and some of them may not even be detected, and this is a great challenge for applications that require the continuation of their services even when the system is under attack or when some of its components have been compromised by an attacker or intruder.

Byzantine-resilient state machine replication is a powerful abstraction that has been widely used to implement systems capable of tolerating arbitrary component failures. That is, in such byzantine-resilient systems, n – t replicated state machines maintain their state consistent despite the action of up to t arbitrarily or byzantine faulty state machines. This notion was extended for tolerating malicious attacks or intrusions so that a system operates correctly even when some of its components get compromised by a malicious intruder or attacker. By combining replica diversity, voting and cryptographic schemes, a byzantine state machine replication based intrusion tolerant system (ITS) can mask a number of compromised replicas, so the system can continue operation without a disruption - with perhaps a degraded performance [24,28,29,31,32,33].

In order to maintain consistent states, replicated state machines must agree on a total order sequence of client operations [15], and many related protocols were motivated by the need to circumvent the consensus impossibility result in asynchronous distributed systems [16], either by adding system model assumptions (e.g., [17,27]) or by weakening the required consensus properties (e.g., [30]). Hence, existing protocols for byzantine state-machine replication work on a variety of system model assumptions and protocol

properties. Though the ideas supporting byzantine state machine replication had been around for a few decades [3,15], it took some time until new protocols overcame the prohibitive performance costs of the first solutions, improving byzantine replication performance in aspects like operations latency, throughput, message overhead and minimum number of required replicas [4,18,19,20,5,25,27]. These optimized solutions have diverse characteristics, and each one represents a distinct trade-off in terms of cost and efficiency where optimizing one feature usually implies in sacrificing another [28]. For instance, an implementation that optimizes agreement, by using a centralized message ordering scheme, may expose the replication protocol to specific adversary attacks [21]. Other protocols that separate the ordering phase from the execution phase to improve resource utilization [23], as fewer replicas area required, are more vulnerable to certain attacks on a specific set of replicas (ordering replicas). Solutions that use specially equipped components, such as trusted components [27], are more efficient in resource and time, but have the whole robustness and cost of the system dependent on the implementation of such trusted components, and so forth.

Although these optimized protocols can also be applied for intrusion tolerance [24, 27,31], one of the main concerns is the conventional non-correlated fault assumption because an intruder can explore vulnerabilities of a compromised replica to attack others. Thus, replica diversity, intrusion detection, reconfiguration and proactive recovery are techniques habitually combined in such settings to implement more robust byzantine state machine replication [22,24,27,28,31].

Because byzantine state machine replication based ITS has additional security concerns when compared to conventional byzantine replication, such as the existence of an intelligent adversary and the need for confidentiality - not only integrity and availability -, performance issues already addressed for fault-tolerant systems become more vital. Attacks to slowdown the system or even denial-of-service attacks are of great concerns. For example, attackers may exploit specific implementation characteristics to slowdown execution in a way that makes the system unusable [21, 26].

In general terms, system robustness depends on a given byzantine replication implementation, level of replica diversity, specific attack detection and reconfiguration mechanisms, recovery strategy and so on. These characteristics may come with increasing complexity and cost, with consequences in performance. Moreover, if such defence mechanisms are triggered too often or if system configuration is not adequate, system performance may be affected to an unacceptable level. Additionally, CPS do not always operate in controlled environments, so unexpected conditions may occur during execution in physical processes as well as in the network.

With the aim of handling such dynamicity, researches have proposed adaptive solutions. For example, in [21] adaptive timeouts have been used to avoid the exploitation of long timeouts that delay protocols steps. The authors in [5] developed a version of byzantine replication [4] where the batch size and batching timeout are regulated by a controller so as to optimize message throughput and delivery time. Other systems have proposed ways to automatically adapt server redundancy to the level of attack alerts [34].

3. Autonomic Byzantine Replication

While adaptive mechanisms are suitable to protect systems against malicious and non-malicious variations in client and system activities, such adaptive mechanisms may also need to be modified to cope with unanticipated changes in the computing environment and/or client/system requirements - such as new system/network

configurations, new system component versions or even new SLAs [36]. Moreover, changes in defence policies may be required: responses to attacks could trigger new modes of operation - for example, aiming at a degraded but safer operation. Thus, adaptation must be on-the-fly and without complete previous knowledge or anticipation of what may occur. To address these issues the system must be equipped with self-manageable or autonomic behavior [6]. That is, the system should change its adaptation policies at runtime, when requited. We called systems with such a capability as *autonomic adaptive*, instead of *online adaptive* or even *offline adaptive*; see [35] for a discussion. Therefore, feedback loops are required for sensing both the environment and system requirements, and systems should dynamically adapt themselves according to such perceived environment and higher level policies.

A few researches have successfully addressed some autonomic properties for intrusion tolerance [32], but developing fully autonomic byzantine-resilience state machine replication remains a challenge. We have effectively applied autonomic or self-adaptive mechanisms to implement some fault tolerant distributed protocols, such as failure detection [7] and group communication [8]. In these autonomic approaches, the protocol is itself an object to be managed by a built-in controller [37]. To design an autonomic byzantine state machine replication protocol, a number of questions must be further addressed, among others: What should be the protocol performance objectives and how to express them? How system and protocol dynamics can be modelled in the loop? When and how to adapt to distinct modes of operations and distinct optimized versions of byzantine replication? How frequently should the system components and protocol variables be monitored and reconfigured?

The Intrusion Tolerant System will produce dynamically defined objectives for the Byzantine Fault-Tolerant Replication (BFT) configuration, according to security alerts, intrusion detections, new modes of operation, resource conditions, and so on (see Figure 1). Such objectives have to be compared against perceived behavior from the BFT sensors, and the calculated error or deviation is passed to the BFT Controller that will apply the appropriated control laws to produce new setups that will adjust or reconfigure the BFT protocol to the intended behavior.

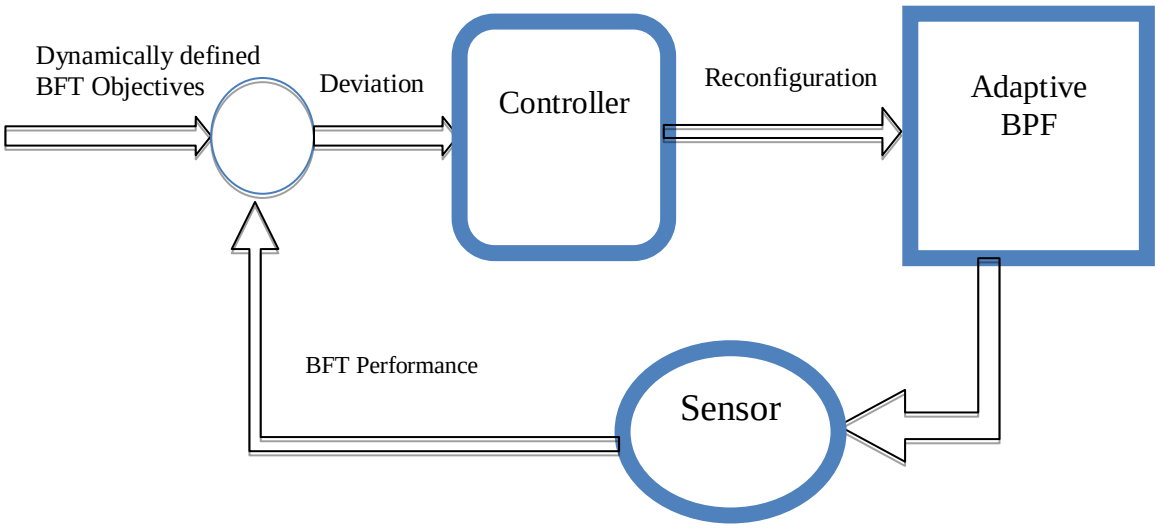


Figure 1. Autonomic Byzantine Replication Loop

As an example, consider that a BFT system must optimize latency and throughput most of the time, except when some urgent message is sent by the upper-layer application. Such urgent message can be used, for instance, to safely halt the system after a detected attack. In such a scenario, the *latency/throughput optimization* policy must be changed to an *urgent delivery* policy. As a motivating example, consider the adaptive byzantine replication protocol presented in [5]. In such a protocol, we used the abstraction of a pipeline with 4 stages: Message checking (to discard invalid or non-authenticated messages); Buffering and Batching (to handle a number of client requests); Ordering (to agree in an order to request processing); Processing (to compute and reply the requests). Our basic idea in such an approach is to keep the pipeline as busy as possible, and two variables are controlled to achieve the best results: batch timeout (BT) and batch size (BS). BT must be long enough to allow for the grouping of a number of requests, and it must be short enough to guarantee at least one batch is in the ordering stage (too long timeouts may make the ordering stage empty). As for BT, larger values will decrease cryptographic and ordering costs, but, on the other hand, may delay message delivery. To constantly optimize throughput, BT and BS are periodically and automatically adjusted to current client activity and protocol/system performance. Such adaptive behaviour although optimizes the relation throughput/delay, it does not help with urgent messages. In the autonomic version (or *autonomic adaptive*, following the terminology presented in [35]), distinct modes of operation must be modelled to allow the automatic modification of adaptive policies. In this simple example, the mode *urgent delivery* must be included, which will result in faster delivery when required – this can be done, for instance, just by setting BT close to zero. We are currently working on the implementation of such a new approach that will be later integrated into a new security framework for IoT-enabled CPS.

4. Final Remarks

In this position paper we discussed the state-of-art of byzantine replication for implementing intrusion tolerance, and argued that autonomic or self-adaptive approaches are required when considering the complexity of IoT-enabled CPS. An earliest sketch of the proposed solution is presented, which is being developed in the Distributed System Laboratory (LaSiD) at UFBA.

References

[1] J. Fraga, and D. Powell, A Fault- And Intrusion-Tolerant File System, in proceedings of IFIP 3rd International Conference on Computer Security, Dublin, Ireland, pp. 203218, 1985.

[2] S. Hossain, S. Etigowni, K. Davis, and S. Zonouz, Towards cyber-physical intrusion tolerance, in proceedings of the IEEE International Conference on Smart Grid Communications (SmartGridComm), Miami, FL, USA, pp. 139-144, 2015.

[3] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem, in ACM Transactions on Programming Languages and Systems. vol. 4, no. 3, pp. 382-401, Jan.-Feb. 2016.

[4] M. Castro and B. Liskov, "Practical Byzantine fault-tolerance and proactive recovery", in ACM Transactions on Computer Systems (TOCS), vol. 20, no. 4, November 2002,

[5] A. Sá, A.Freitas, and R.Macêdo, Adaptive request batching for byzantine replication, in Operating Systems Review, vol. 47, no. 1., pp. 35-42, 2013

[6] J. Kephart and D. Chess. The vision of autonomic computing, in IEEE Computer, vol. 36, no. 1, pp. 41-50, 2003.

[7] A. Santos Sá, R, José, and A. Macêdo. QoS Self-configuring Failure Detectors for Distributed Systems. Frank Eliassen;; Rüdiger Kapitza. Distributed Applications and Interoperable Systems, 6115, Springer Lecture Notes in Computer Science, pp.126-140, 2010.

[8] R. Macêdo, A. Freitas, and A. Sá. Enhancing group communication with self-manageable behavior. Journal of Parallel and Distributed Computing. vol. 73, no. 4. pp. 420-433, April 2013

[9] S. Andrade, and R. Macêdo, Architectural design spaces for feedback control concerns in self-adaptive systems. in Proceedings of the 25th International Conference on Software Engineering and Knowledge Engineering (SEKE), Boston, USA, pp. 741-746, June, 2013

[10] S. Andrade, and R. Macêdo. A non-intrusive component-based approach for deploying unanticipated self-management behavior, in Proceedings of ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '09), pp. 152-161, June 2009.

[11] E. Lee. Cyber-Physical Systems: Design challenges, in Proceedings of the 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'2008), Orlando, FL, pp. 363-369, 2008.

[12] Y. Deswarte, L. Blain, J., and C. Fabre, Intrusion tolerance in distributed computing systems, In proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy. pp. 110-121, 1991.

[13] M. Huebscher, and J. A. Mccann, A survey of autonomic computing—degrees, models, and applications. ACM Computing Surveys (CSUR), vol. 40, no. 3, p. 7, 2008.

[14] I. Atzori, A. Iera, and G. Morabito, The internet of things: A survey, in Computer networks, vol. 54, no. 15, pp. 2787-2805, 2010.

[15] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," ACM Computing Surveys, vol. 22, pp. 299–319, Dec. 1990

[16] M. J. Fischer, N. A. Lynch, and M. S. Paterson. 1985. Impossibility of distributed consensus with one faulty process. *J. ACM* 32, 2 (April 1985), 374-382.

[17] C. Dwork, N. Lynch, and L. Stockmeyer. 1988. Consensus in the presence of partial synchrony. *J. ACM* 35, 2 (April 1988), 288-323.

[18] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: speculative byzantine fault tolerance," in Proc. of 21th ACM SIGOPS Symp. on Operating systems principles. ACM, 2007, pp. 45–58.

[19] B. Wester, J. Cowling, E. Nightingale, P. Chen, J. Flinn, and B. Liskov, "Tolerating latency in replicated state machines through client speculation," in Proc. of the 6th USENIX Symp. on Networked systems design and implementation (NSDI). USENIX, April 2009, pp. 245–260.

[20] R. Guerraoui, N. Knezevi˘c,´ V. Quema,´ and M. Vukolic,´ "The next 700 BFT protocols," in Proc. of the 5th European Conf. on Computer systems (EuroSys). ACM, April 2010, pp. 363–376.

[21] Y. Amir, B. Coan, J. Kirsch and J. Lane, "Byzantine replication under attack," 2008 IEEE Intl Conference on Dependable Systems and Networks (DSN), Anchorage, AK, 2008, pp. 197-206.

[22] D. Malkhi and M. Reiter, "Unreliable intrusion detection in distributed computations," in Proc. of the 10th Computer Security Foundations Workshop (CSFW). IEEE CS, June 2002, pp. 116–124.

[23] J. Yin, J. Martin, A.Venkataramani, L. Alvisi, and M. Dahlin. 2003. Separating agreement from execution for byzantine fault tolerant services. SIGOPS Oper. Syst. Rev. 37, 5 (October 2003), 253-267.

[24] Q. Nguyen and A. Sood, "A comparison of intrusion-tolerant system architectures," IEEE Security & Privacy, vol. 9, no. 2, pp. 18–25, Mar./Apr. 2003.

[25] T. Wood, R. Singh, A. Venkataramani, P. Shenoy, and E. Cecchet, "ZZ and the art of practical BFT execution," in Proceedings of the 6th ACM SIGOPS/EuroSys European Systems Conference EuroSys'11, Apr. 2011.

[26] P. Sousa, N. F. Neves, and P. Veríssimo, "How resilient are Distributed fault/intrusion-tolerant systems?" in Proceedings of Dependable Systems and Networks – DSN 05, Jun. 2005, pp. 98–107.

[27] P. Sousa, A. N. Bessani, M. Correia, N. F. Neves, and P. Verissimo, "Highly available intrusion-tolerant services with proactive-reactive recovery,"IEEE Transactions on Parallel and Distributed Systems, vol. 21, no. 4, pp. 452–465, 2010.

[28] F. Wang, R. Uppalli and C. Killian, "Analysis of techniques for building intrusion tolerant server systems," IEEE Military Communications Conference, 2003. MILCOM 2003., 2003, pp. 729-734 Vol.2.

[29] S. Heo, P. Kim, Y. Shin, J. Lim, D. Koo, Y. Kim, O. Kwon, and H. Yoon. A Survey on Intrusion-Tolerant System JCSE, vol. 7, no. 4, pp.242-250, 2013.

[30] M. O. Rabin, "Randomized Byzantine generals," in Proc. 24th IEEE Symposium on Foundations of Computer Science, pp. 403–409, 1983,

[31] B. Foo et al.,. Intrusion Response Systems: A Survey": Book chapter in "Information Assurance: Dependability and Security in Networked Systems", pp. 377-416, Morgan Kaufmann Publishers. 2007.

[32] E. Yuan, N. Esfahani, and S. Malek. 2014. A Systematic Survey of Self-Protecting Software Systems. ACM Trans. Auton. Adapt. Syst. 8, 4, Article 17 (January 2014), 41 pages.

[33] A. Valdes et a., An Architecture for an Adaptive Intrusion-Tolerant Server. Volume 2845, Lecture Notes in Computer Science pp 158-178.

[34] G. Schirner, D. Erdogmus, K. Chowdhury and T. Padir, The Future of Human-in-the-Loop Cyber-Physical Systems, in IEEE Computer, vol. 46, no. 1.

[35] Macêdo, R. J. de A. A Vision on Autonomic Distributed Systems. WoSiDA 2013.

[36] Macêdo, Raimundo J. de A. "Approaches for Adaptive and Dependable Distributed Systems" (Invited Talk)[slides: http://webhost.laas.fr/TSF/IFIPWG/Workshops&Meetings/53/workshop/11.Macedo.pdf]. Workshop on Dependability of Large-Scale and Dynamic Systems,53rd Meeting of IFIP Working Group 10.4 - Dependable Computing and Fault Tolerance, Natal, Rio Grande do Norte, Brazil, February 21-25, 2008.

[37] Macêdo, Raimundo J. de A. "Self-Manageable Protocols for Dependability" (Invited Talk) http://webhost.laas.fr/TSF/IFIPWG/Workshops&Meetings/64/WorkshopregularPapers/SESSION%204/macedo-ifip%20workshop%2013v3.pdf]. Workshop on Dependability and Fault Tolerance. 64rd Meeting of IFIP Working Group 10.4 - Dependable Computing and Fault Tolerance. Visegrád, Hungary June 27-30, 2013.

VII Workshop On Autonomic Distributed Systems (WoSiDa)

SBRC 2017

Sessão Técnica 2

Uma Abordagem de Projeto para Mecanismos Autonômicos de Tolerância a Falhas em Sistemas Distribuídos

Alirio Santos de Sá[1,2], **Raimundo José de Araújo Macêdo**[1,2]

[1]Laboratório de Sistemas Distribuídos (LaSiD), Departamento de Ciência da Computação, Instituto de Matemática, Universidade Federal da Bahia, Bahia, Brasil

[2]Pró-reitoria de Tecnologia de Informação e Comunicação (PROTIC) Universidade Federal do Sul da Bahia (UFSB), Bahia, Brasil

{aliriosa,macedo}@ufba.br

Abstract. *Modern distributed system are characterized by compositions, resource provisioning and application requirements that can change over time. In such a systems, traditional fault-tolerance mechanisms are inefficient to address application performance requirements while at the same time addressing dependability. Even traditional adaptive fault-tolerant mechanisms cannot succeed because their adaptation approaches rely on requirements and environment behaviors defined during the design. In this paper, we discuss a design approach to build autonomic fault-tolerant mechanisms, which are capable of self-configuring according to changes in the computing environment and in user-defined requirements. As a proof of concept, we discuss the implementation of two basic autonomic fault-tolerance mechanisms.*

1. Introdução

Mecanismos de tolerância a falhas são fundamentais para dotar os ambientes e serviços distribuídos da redundância necessária para que os mesmos possam *mascarar* ou *compensar* a ocorrência de falhas [Cristian 1991]. Entretanto, para cumprirem seus objetivos, tais mecanismos demandam custos computacionais adicionais, em termos de processamento e comunicação [Jalote 1994]. Assim, para não comprometerem o funcionamento normal dos sistemas, os mecanismos de tolerância a falhas devem ser adequadamente configurados, de modo a conciliar estes custos adicionais com a expectativa de carga das aplicações, com os requisitos de desempenho especificados e com o grau desejado de confiabilidade [Cristian 1991]. Para assegurar a eficiência e eficácia, atividades de gerenciamento devem ser realizadas – i.e. monitoramento contínuo dos sistemas e dos mecanismos de tolerância a falhas e, também, ajustes ou re-configuração de parâmetros, quando as condições do ambiente ou os requisitos das aplicações divergem dos considerados no projeto.

Novas facilidades de processamento e comunicação propiciam o surgimento de ambientes distribuídos dinâmicos, representados por aplicações com características e requisitos dinâmicos e também por plataformas computacionais que permitem não apenas a alocação, liberação e migração dinâmica de recursos virtualizados, mas também a definição de composições dinâmicas, em que os componentes são definidos em tempo de execução (a partir da entrada e saída espontânea de dispositivos e processos, por exemplo). Casos típicos desses ambientes distribuídos são encontrados, por exemplo, em plataformas de P2P (Peer-to-Peer), de grades computacionais e de computação em nuvens.

Por um lado, estes ambientes dinâmicos trazem benefícios, pois permitem que os recursos sejam ajustados, em tempo de execução, às necessidades das aplicações e facilitam a implementação de estratégias de reconfiguração dinâmica, necessárias em caso de falhas, por exemplo. Todavia, esta dinamicidade implica em novas componentes de incerteza nos ambientes distribuídos, pois a carga computacional, os recursos computacionais disponíveis e os requisitos e características das aplicações variam ao longo do tempo.

Por outro lado, ambientes com características dinâmicas dificultam o projeto e a gestão da maioria dos mecanismos de tolerância a falhas, os quais usam estimativas *off-line* a respeito dos recursos e dos requisitos e características das aplicações para definirem adequadamente seus parâmetros operacionais. Além disso, muitos dos mecanismos adaptativos de tolerância a falhas existentes na literatura não estão preparados para lidar com esta dinamicidade, uma vez que não consideram, na adaptação de seus parâmetros operacionais, mudanças dinâmicas nas características do ambiente, na carga computacional ou nos requisitos das aplicações. Mais ainda, muitas das suposições usadas por estes mecanismos adaptativos, sobre o comportamento do ambiente, não são apropriadas – e.g. comportamentos probabilísticos específicos. Neste contexto, propusemos, em trabalhos anteriores[1], mecanismos auto-gerenciáveis (autonômicos) de tolerância a falhas para lidar com o desafio da implementação de estratégias que permitam que os mecanismos tradicionais de tolerância a falhas possam suportar a confiabilidade, adaptando dinamicamente seus parâmetros de configuração a partir das mudanças observadas nas características dos ambientes distribuídos e nos requisitos de qualidade de serviço das aplicações.

Baseado nessas experiências e nas discuções introduzidas em [Macêdo 2008] e em [de Sá 2011], a proposta deste artigo é discutir uma estratégia para o projeto e implementação de mecanismos autonômicos de tolerância a falhas. Para isto, apresentamos uma metodologia para a implementação do laço de gestão, abordamos como os mecanismos tradicionais de tolerância a falhas podem ser especializados para apresentarem a habilidade de auto-gestão, comentamos como a teoria de controle de sistemas dinâmicos pode ser usada na construção dos mecanismos autonômicos de tolerância a falhas e apontamos, como estudo de caso, a implementação da facilidade de gestão autonômica em dois mecanismos básicos para construção de sistemas distribuídos confiáveis.

2. A Abordagem de Projeto Proposta

A estratégia de projeto proposta visa atender aos seguintes requisitos: (a) *Auto-gestão* – enfrentar aos desafios impostos pelos ambientes distribuídos modernos, a partir da proposição de mecanismos de tolerância a falhas com habilidade de auto-gestão; (b) *Suporte a requisitos dinâmicos de qualidade de serviço* – atender a requisitos dinamicamente definidos pelas aplicações ou usuários; (c) *Suporte ao legado de soluções existentes* – permitir que os mecanismos autonômicos propostos aproveitem as facilidades dos mecanismos de tolerância a falhas existentes, sem comprometer a correção (i.e. *safety* e *liveness*[Lamport and Lynch 1989]) dos mesmos; (d) *Reuso* – conceber estratégias com baixo acoplamento em relação ao ambiente de execução, podendo atuar em diferentes modelos de sistemas distribuídos; (e) *Modelagem do comportamento dinâmico* – explorar estratégias da literatura que permitam a análise e síntese de modelos para lidar com o comportamento dinâmico do ambiente distribuído, sem comprometer o reuso da solução.

[1] e.g., [de Sá and Macêdo 2010], [de Sá et al. 2013], [Macêdo et al. 2013], [de Sá et al. 2014] etc.

A abordagem de projeto proposta considera a configuração dos parâmetros dos mecanismos de tolerância a falhas de modo a permitir que o ajuste das características destes mecanismos sejam levados para um nível de abstração mais próximo das aplicações – seguindo a abordagem da computação autonômica [Horn 2001]. Isto implica em um novo modelo de alocação de responsabilidades, no qual as aplicações ou os usuários devem definir, em tempo de execução, as suas demandas ou restrições em termos de requisitos de qualidade de serviço (e.g. tempo de resposta, confiabilidade, percentual de uso de recursos etc.). Os mecanismos autonômicos de tolerância a falhas, por sua vez, devem observar as variações dinâmicas nas características do ambiente e ajustar seus parâmetros operacionais de modo a atender os requisitos dinamicamente definidos.

Software baseado em Camadas para Obter Correção e Reuso. Conceitualmente, os mecanismos autonômicos propostos podem ser vistos como uma estratégia que provê o suporte à confiabilidade, encapsulando os mecanismos tradicionais ou adaptativos de tolerância a falhas (ver Figura 1). Nesta abordagem, os mecanismos tradicionais de tolerância a falhas se preocupam com o suporte a confiabilidade, garantindo as propriedades de correção (*safety* e *liveness*) de seu serviço, a partir de parâmetros de configuração fixados durante o projeto. Em uma segunda camada, estratégias adaptativas encapsulam (ou especializam) as facilidades dos mecanismos tradicionais para adaptar as configurações, considerando suposições (ou hipóteses) fixadas durante o projeto – e.g. distribuições de probabilidade, arranjo dos componentes, disponibilidade dos recursos etc. Por fim, mecanismos autonômicos de tolerância a falhas encapsulam os mecanismos adaptativos, permitindo a auto-configuração quando as características do ambiente ou requisitos das aplicações divergem do que foi definido em projeto.

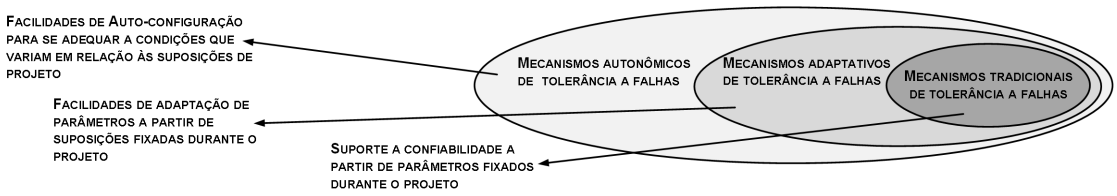


Figura 1. Visão Conceitual: Modelagem baseada em Camada

Arquitetura Autonômica para Implementar Auto-Gestão. A concepção da habilidade de auto-gerenciamento (i.e. auto-configuração) requer que os mecanismos de tolerância a falhas implementem um laço de gerenciamento autonômico [Horn 2001], envolvendo monitoramento, planejamento e intervenções – ver Figura 2.

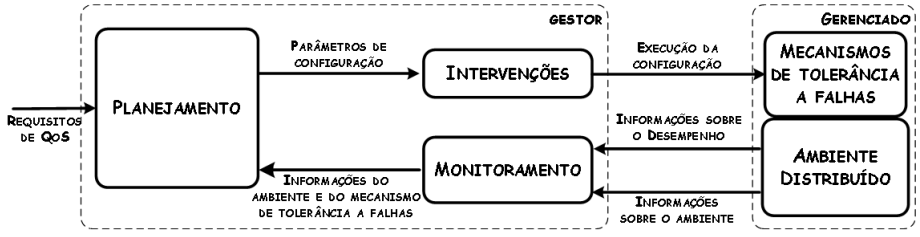


Figura 2. Laço de gestão implementado pelos mecanismos de tolerância a falhas

O monitoramento envolve a coleta de informações associadas às características dinâmicas do ambiente e à qualidade de serviço entregue pelos mecanismos de tolerância

a falhas. O planejamento envolve a definição, em tempo de execução, dos parâmetros operacionais adequados para levar o desempenho do mecanismo de tolerância a falhas a atender os requisitos de qualidade de serviço definidos. O cálculo dos valores dos parâmetros também leva em consideração o estado atual do ambiente distribuído. As intervenções envolvem a implantação, nos mecanismos de tolerância a falhas, dos parâmetros operacionais definidos durante o planejamento.

Teoria de Controle para Modelar o Comportamento Dinâmico do Sistema. O laço de gestão autonômico se assemelha aos laços de controle implementados, no campo da engenharia, na automação e no controle de sistemas industriais [Ogata 1995]. Como exemplo, pode-se fazer o paralelo entre um mecanismo autonômico de detecção de defeitos e um sistema de controle de motor DC (corrente contínua), ver Figura 3.

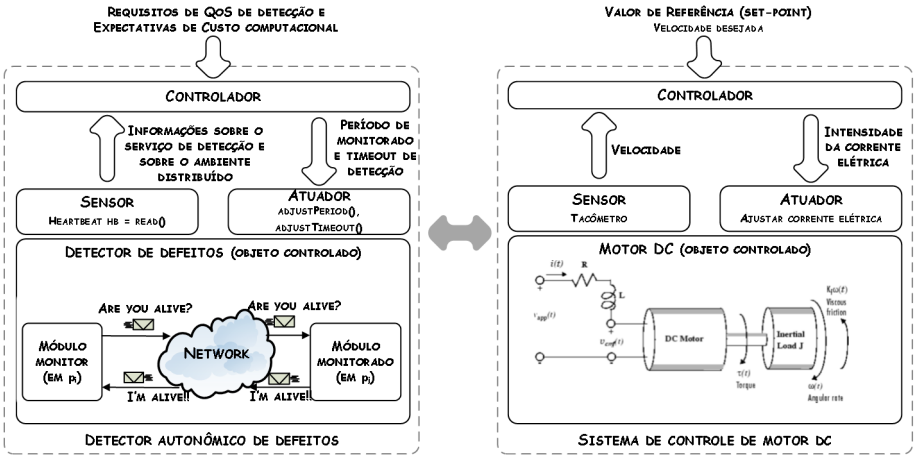


Figura 3. Comparativo entre detector autonômico e sistema de controle de motor

Nesse exemplo, um tacômetro (sensor) monitora a velocidade do motor (objeto controlado). Em seguida, um controlador verifica se a velocidade observada está de acordo com um valor de referência (*set-point*). Existindo desvios entre as velocidades, determina um novo valor de corrente elétrica (configuração) para o motor DC, de modo que o mesmo apresente a velocidade desejada. Então, um atuador ajusta a corrente elétrica do motor, usando o valor informado pelo controlador. No caso do detector, métodos de leitura (sensor) coletam informações sobre comportamento do ambiente e o desempenho do serviço de detecção (objeto controlado), usando mensagens de monitoramento, por exemplo. Em seguida, o controlador verifica se o desempenho observado está de acordo com os requisitos de qualidade de serviço de detecção e as expectativas de custo definidas pelo usuário (i.e. *set-points*). Existindo divergências, o controlador determina os novos parâmetros do detector (e.g. período de monitoramento de falhas e *timeout* de detecção), de modo que o detector apresente o desempenho desejado em termos de qualidade de serviço de detecção e custo computacional. Em seguida, o controlador ativa métodos de escrita (atuador) necessários para implantar os novos parâmetros de configuração.

Assim, na perspectiva do projeto, o laço de gestão dos mecanismos autonômicos de tolerância a falhas é projetado seguindo a mesma abordagem usada nos sistemas de controle, isto é: embutindo, nos mecanismos de tolerância a falhas existentes, controladores (ou gestores autonômicos) responsáveis pelo planejamento das configurações, além de métodos de sensoriamento e atuação que realizam a coleta das informações e

a implantação dos novos parâmetros de configuração, respectivamente. Por conta das técnicas de análise e síntese disponibilizadas pela teoria de controle de sistemas dinâmicos [Hellerstein et al. 2004], a implementação dos mecanismos de tolerância a falhas propostos utilizam tal ferramental matemático para a modelagem de suas funções autonômicas.

3. Estudos de Caso: Detecção de Defeitos e Comunicação em Grupo

A estratégia de projeto proposta foi utilizada na implementação de detectores de defeitos e protocolos de comunicação em grupo em sistemas distribuídos sujeitos a falhas por crash (parada). Detectores de defeitos são elementos básicos na implementação da confiabilidade em sistemas distribuídos, seja para permitir a ativação de procedimentos de recuperação, seja para possibilitar o acionamento de mecanismos de reconfiguração na ocorrência de falhas. Protocolos de comunicação em grupo são fundamentais para implementação de coordenação distribuída e sincronização de estados em serviços replicados – um esquema comum em tolerância a falhas.

O detector autonômico de defeitos, proposto em [de Sá and Macêdo 2010], foi o primeiro a demonstrar como parâmetros de detecção (i.e. período de monitoramento e *timeout* de detecção) podem ser configurados em tempo de execução, considerando métricas de qualidade de serviço de detecção de defeitos [Chen et al. 2002] e mudanças nas características do ambiente. Para isto, este abordagem encapsula um detector adaptativo de defeitos de modo a dotá-lo da capacidade de auto-configuração. O gestor autonômico da abordagem é implementado a partir de controladores PID [Ogata 1995], os quais recebem da aplicação os requisitos em termos de limite de consumo de recursos e qualidade de serviço (QoS) de detecção e, baseado no desempenho do detector, monitorado em um dado instante, em termos da QoS de detecção e do consumo de recursos do detector, sugerem novos valores para os parâmetros do detector.

O protocolo autonômico de comunicação em grupo, proposto em [Macêdo et al. 2013], foi o primeiro a demonstrar como atender a relação de compromisso entre custo (i.e. sobrecarga de mensagens) e latência de entrega (i.e. tempo de resposta), considerando requisitos definidos pelo usuário, i.e. percentual de consumo de recursos. Para isto, encapsula um protocolo de comunicação em grupo adaptável, de modo a dotá-lo da capacidade de auto-configuração. O gestor autonômico da abordagem é implementado a partir de dois laços de controle em série e ajusta os parâmetros do protocolo para: usar mais recursos e entregar mensagens mais rapidamente, quando o percentual de consumo de recursos estiver abaixo do definido pelo usuário; ou reduzir o uso de recursos em cenários de sobrecarga no ambiente computacional.

4. Considerações Finais

Neste artigo, discutimos uma abordagem para a construção de mecanismos autonômicos de tolerância a falhas. A estratégia proposta combina software baseado em camadas, laços autonômicos e teoria de controle realimentado com o intuito de garantir a correção do funcionamento do mecanismo, de possibilitar reuso das facilidades de mecanismos de tolerância a falhas tradicionais existentes e de implementar auto-configuração de parâmetros em ambientes dinâmicos. Para demonstrar a viabilidade, a abordagem apresentada foi usada na construção de mecanismos autonômicos de tolerância a falhas – ver [de Sá and Macêdo 2010] e [Macêdo et al. 2013]. Além disso, outros mecanismos au-

tonômicos de tolerância a falhas também estão sendo construidos usando a abordagem proposta – e.g., [de Sá et al. 2014] e [Moreira and de Sá 2014].

Referências

Chen, W., Toueg, S., and Aguilera, M. K. (2002). On the quality of service of failure detectores. *IEEE Transactions on Computers*, 51(2):561–580.

Cristian, F. (1991). Understanding fault-tolerant distributed systems. *Comm. of the ACM*, 34(2):56–78.

de Sá, A. S. (2011). *Mecanismos Autonômicos de Tolerância a Falhas para Sistemas Distribuídos*. PhD thesis, Pós-Graduação em Ciência da Computação, Universidade Federal da Bahia.

de Sá, A. S., Freitas, A. E. S., and Macêdo, R. J. A. (2013). Adaptive request batching for byzantine replication. *SIGOPS Oper. Syst. Rev.*, 47(1):35–42.

de Sá, A. S. and Macêdo, R. J. A. (2010). Qos self-configuring failure detectors for distributed systems. In *Proc. of the 10th IFIP WG 6.1 Intern. Conf. on Distributed Applications and Interoperable Systems (DAIS' 2010)*, volume 6115 of *Lecture Notes in Computer Science (LNCS)*, pages 126–140, Berlin, Heidelberg. Springer-Verlag.

de Sá, A. S., Macêdo, R. J. A., and Queiroz, H. S. (2014). Protocolos auto-gerenciáveis de consenso para sistemas distribuídos parcialmente síncronos. In *Anais do IV Workshop de Sistemas Distribuídos Autonômicos (WoSiDA'2014) do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 7–10. SBC.

Hellerstein, J. L., Diao, Y., Parekh, S., and Tilbury, D. M. (2004). *Feedback control of computing systems*. Wiley-Interscience, Canada.

Horn, P. (2001). Autonomic computing: Ibm's perspective on the state of information technology. *Computing Systems*, 15(January):1–40.

Jalote, P. (1994). *Fault tolerance in distributed systems*. Prentice Hall, New Jersey.

Lamport, L. and Lynch, N. (1989). Chapter on distributed computing: Methods and models. Technical Report MIT/LCS/TM-384, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA.

Macêdo, R. J. A. (2008). Approaches for adaptive and dependable distributed systems (invited talk). In *Workshop on Dependability of Large-Scale and Dynamic Systems, 53rd Meeting of IFIP Working Group 10.4 - Dependable Computing and Fault Tolerance*, Natal, RN, Brazil [slides: http://webhost.laas.fr/TSF/IFIPWG/Workshops&Meetings/53/workshop/11.Macedo.pdf].

Macêdo, R. J. A., Freitas, A. E. S., and de Sá, A. S. (2013). Enhancing group communication with self-manageable behavior. *Journal of Parallel and Distributed Computing*, 73(4):420 – 433.

Moreira, F. L. and de Sá, A. S. (2014). Um mecanismo de auto-gestão para sistemas de armazenamento de dados replicados nas nuvens. In *Anais do IV Workshop de Sistemas Distribuídos Autonômicos (WoSiDA'2014) do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 11–14. SBC.

Ogata, K. (1995). *Discrete-time control systems*. Prentice-Hall, USA, 2nd edition.

Um Protocolo Cooperativo para Acesso ao Meio em Redes de Sensores Aquáticas Sem Fio.

Lucas S. Cerqueira[1], Felipe R. da Silva[1], Luiz Filipe M. Vieira[2], Marcos Augusto M. Vieira[2], José Augusto M. Nacif[3], Alex B. Vieira[1],

[1]Departamento de Ciência da Computação
Universidade Federal de Juiz de Fora – Juiz de Fora, MG – Brasil.

[2]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais – Belo Horizonte, MG – Brasil.

[3]Instituto de Ciências Exatas e Tecnológicas
Universidade Federal de Viçosa – Florestal, MG – Brasil.

lucas.saar@ice.ufjf.br, {alex.borges, felipe.rooke}@ufjf.edu.br,

{lfvieira, mmvieira}@dcc.ufmg.br, jnacif@ufv.br

Abstract. *Wireless communication in aquatic environments poses great challenges, since electromagnetic and optical waves do not propagate for long distances in the environment. In this scenario, an acoustic channel becomes a better option, even though bringing challenges as a higher latency, lower bandwidth and higher error rate. To improve the communication quality, this work proposes a MAC layer protocol based on the time division multiple access (TDMA) scheme and that uses cooperation, thereby taking advantage of the idleness of the nodes to retransmit messages that have failed. Our protocol has been evaluated on the ns-3 simulator and, the results show a significant improvement in the network metrics. For example, the average packet loss rate is reduced by up to 26% and goodput increases by up to 7.36%.*

Resumo. *A comunicação sem fio em ambientes aquáticos impõe grandes desafios, uma vez que ondas eletromagnéticas e ópticas não se propagam por longas distâncias nesse meio. Nesse cenário, um canal acústico se torna uma melhor opção, ainda que trazendo desafios como uma maior latência na propagação do sinal, largura de banda menor e alta taxa de erros. De forma a melhorar a qualidade na comunicação, este trabalho propõe um protocolo em camada MAC baseado no esquema de acesso múltiplo por divisão de tempo (TDMA) e que utiliza cooperação, aproveitando assim a ociosidade dos nós para retransmissão de mensagens que falham. O protocolo proposto foi avaliado por simulações no ns-3 e, os resultados obtidos mostram que há uma melhora significativa nas métricas de rede avaliadas. Por exemplo, a taxa de perda de pacotes, na média, se reduz em até 26% e o goodput aumenta em até 7,36%.*

1. Introdução

As redes de sensores aquáticos (RSAs) tem se mostrado uma importante área de estudo e sua aplicação é cada vez mais frequente. O uso dessas redes em aplicações militares, comerciais e até mesmo em pesquisa tornam factíveis um número de tarefas até então

desafiadoras. Por exemplo, com o uso de RSAs, é possível realizar com eficiência o monitoramento da vida aquática e da qualidade da água, a exploração de recursos naturais como petróleo e gás, a detecção de minas e veículos submarinos e até mesmo, prevenção de desastres como tsunamis e derramamento de petróleo [Felemban et al. 2015].

As condições do meio aquático impõem grandes dificuldades na comunicação de dados. Na água, ondas eletromagnéticas e ópticas sofrem alta atenuação, sendo absorvidas em poucos metros [Brekhovskikh 2003]. Logo, redes baseadas em ondas de rádio não são apropriadas para o meio aquático. Assim, ondas acústicas têm sido adotadas como padrão tanto na indústria quanto na área acadêmica. No entanto, o canal acústico é caracterizado por três grandes desafios: largura de banda limitada e dependente da distância, o desvanecimento multi-caminho variado pelo tempo e a baixa velocidade do som na água [Heidemann et al. 2012]. Além disso, RSAs possuem energia limitada, o que torna o consumo de energia fator crucial no desenvolvimento de novas tecnologias nessa área.

Para melhorar a qualidade da comunicação e superar esses desafios, muitas técnicas vêm sendo estudadas em roteamento, protocolos MAC e camada física. Uma dessas técnicas é a transmissão cooperativa de mensagens que tenta aproveitar a ociosidade dos nós para retransmitir mensagens que falharam [Dianati et al. 2006]. De forma breve, uma transmissão em meio aquático é feita por *broadcast*. Assim, nós intermediários que escutam uma transmissão que falhou podem servir como cooperadores (*relays*) e retransmitir a mensagem. Assim, o nó que originou a mensagem pode tentar ir adiante, deixando a tarefa de retransmissão com seus parceiros de cooperação. Ou então, ambos podem tentar retransmitir a mesma mensagem, aumentando assim a diversidade de caminhos de transmissão. Enquanto a primeira abordagem aumenta principalmente a vazão, a segunda diminui a taxa de erros.

Neste trabalho é proposto um protocolo com cooperação na camada MAC. Para tal utilizamos o esquema de acesso múltiplo por divisão de tempo (TDMA). O protocolo funciona a partir da sinalização de quais mensagens falharam na última rodada de transmissão. A partir daí, os nós que tenham recebido estas mensagens com sucesso, e que estão ociosos, podem retransmití-las no lugar dos nós originais.

Muitos trabalhos utilizam a cooperação de maneira semelhante a um roteamento, selecionando o melhor *relay* para transmitir sua mensagem [Cheng et al. 2012, Carbonelli and Mitra 2006]. Em nosso trabalho, a cooperação é um mecanismo de recuperação de falhas. Um esquema semelhante é proposto por [Lee et al. 2010], permitindo a retransmissão de mensagens que falharam, porém o controle de acesso ao meio não é considerado. A principal diferença em relação a cooperação é a escolha do *relay* que, em nosso esquema os *relays* se elegem através de mensagens de cooperação WTC ao invés de serem escolhidos pelo nó destino.

O novo protocolo foi avaliado por simulações no ns-3. Foram avaliados cenários compostos por 16 nós que tentam realizar uma comunicação com um nó destino. Em comparação com um cenário sem cooperação, os resultados obtidos mostram que há uma melhora significativa nas métricas de rede avaliadas. Por exemplo, a taxa de perda de pacote, na média, se reduz em até 26%, o *goodput* aumenta em até 7,36% sendo que energia total gasta por pacote entregue com sucesso se reduz em até 6,81%.

2. Formulação do Problema

No meio aquático, a comunicação apresenta alta taxa de erros, baixa vazão e baixa largura de banda. Isso caracteriza de forma única RSAs [Vieira et al. 2010]. Em especial, o problema abordado neste trabalho é a baixa eficiência de transmissão de dados no canal acústico: baixa vazão, alto consumo de energia e alta perda de pacotes.

Neste trabalho consideramos uma rede de sensores aquáticos com saltos únicos composta por N nós e um nó destino. Nesta rede todos os pacotes de dados gerados são endereçados para o nó destino. Consideramos também um esquema de acesso ao meio baseado em divisão por tempo (TDMA). Tipicamente, no protocolo TDMA, a cada nó é atribuído um período de tempo, denominado *slot*, no qual o acesso ao meio é exclusivo. Um nó é dito ocioso quando, em seu *slot*, ele não possui dados a serem transmitidos. Tais períodos de ociosidade podem ser utilizados para a retransmissão de mensagens que falharam de outros nós, o que chamamos –de forma ampla– de cooperação.

No meio aquático podemos estimar a probabilidade de entrega de pacotes baseado na frequência f de transmissão, na modulação da onda acústica, na distância d entre os nós transmissor e receptor [Vieira et al. 2010] e na quantidade de ruído. A perda na potência do sinal em caminho devido ao desvanecimento em larga escala é dada por $A(d, f) = d^k a(f)^d$, onde k é o fator de dispersão que, para um cenário prático, é dado por 1,5. $a(f)$ é o coeficiente de absorção dado pela fórmula de Thorp em [Brekhovskikh 2003]. Com isso calculamos a razão entre sinal-ruído $\gamma(d) = SL - A(d, f) - NL + DI$, onde SL é a potência do sinal de transmissão, NL é o barulho ambiente dado pela equação de Wenz em [Wenz 1962], e DI é o fator diretivo que para hidrofones (e.g. modens acústicos) omnidirecionais $DI = 0$ [Wang et al. 2016].

Podemos então descrever a probabilidade de erro no bit, para a modulação BPSK, dada uma distância d [Rappaport 2002] como: $p_e(d) = \frac{1}{2}\left(1 - \sqrt{\frac{\Gamma(d)}{1+\Gamma(d)}}\right)$, onde $\Gamma(d)$ é dado por $\Gamma(d) = 10^{\gamma(d)/10}$.

3. Protocolo MAC cooperativo para redes aquáticas

No esquema TDMA, cada nó da rede recebe um *slot* de igual tamanho de tempo. Quando o último nó termina seu *slot*, é reiniciado o *slot* do primeiro nó, completando um ciclo. O período que compreende os *slots* do primeiro ao último nó é chamado de *frame*. Em um *frame*, cada nó da rede recebe um *slot* para que possa transmitir mensagens. O *slot* do nó destino é utilizado para sinalizar quais mensagens do *frame* atual falharam, enviando uma única mensagem chamada NACK. A figura 1 ilustra a operação de uma rede com três nós: fonte (F), *relay* (R) e destino (D). Neste exemplo o nó fonte possui duas mensagens a serem transmitidas para D: m_1 e m_2. No *frame* i, o nó fonte transmite m_1 mas a mensagem falha, então o nó destino transmite um NACK para sinalizar o reenvio. No *frame* $i + 1$, F retransmite m_1 e no *frame* $i + 2$, transmite m_2.

Sem perda de generalidade, propomos uma modificação do TDMA onde cada *frame* possui um período de sinalização (SP) e um período de dados (DP). O período de sinalização é utilizado para a troca de mensagens de controle de cooperação (WTC). No período de dados ocorre a troca de mensagens como no TDMA tradicional. A figura 2 ilustra o mesmo cenário da figura 1, porém agora, com cooperação. No *frame* i, o nó R

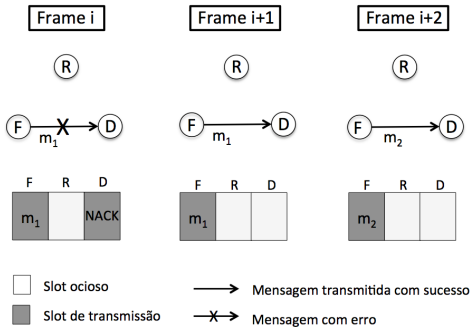


Figura 1. Exemplo de TDMA sem cooperação

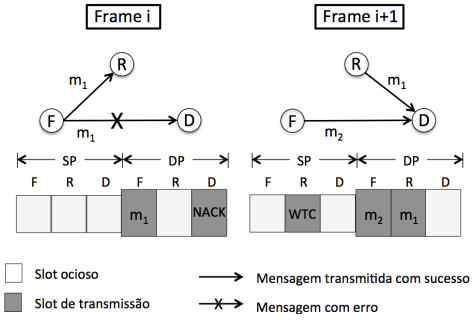


Figura 2. Modo de operação com cooperação

recebe m_1 com sucesso e, logo em seguida, o NACK transmitido por D. No próximo *frame* R está ocioso e possui a mensagem m_1, logo, sinaliza que vai cooperar em seu próximo *slot* enviando a mensagem WTC. Assim, o nó F pode transmitir sua próxima mensagem m_2 no mesmo *frame* que R retransmite m_1. Observe que as mesmas duas mensagens foram enviadas utilizando dois *frames* ao invés de três, como no exemplo da figura 1.

A escolha do *relay* é crucial para o sucesso da cooperação. Em nossa abordagem, ordenamos os *slots* dos *relays* em relação a sua distância do nó destino, semelhante ao que foi feito em [Domingo 2011]. Assim, os primeiros *slots* de cada *frame* serão os dos *relays* mais próximos do destino, dando-os maior oportunidade de se tornarem nós cooperadores. O primeiro nó a enviar a mensagem WTC será escolhido como cooperador. Estes nós são priorizados por possuírem uma menor taxa de erro de pacote em relação aos nós mais distantes, devido a taxa de erro de pacote ser proporcional a distância, de acordo com a equação de probabilidade de erro no bit apresentada da seção 2. Estes nós transmitem mais mensagens que os outros *relays*, causando um consumo desigual de energia.

4. Avaliação

4.1. Metodologia de avaliação

Para avaliar o protocolo proposto, utilizamos o ns-3[1], um simulador de redes baseado em eventos discretos. Implementamos no ns-3 o protocolo proposto juntamente com o TDMA tradicional e a equação de probabilidade de erro no bit apresentada na seção 2. O cenário de simulação é composto por 16 nós, sendo 10 nós fontes, 5 nós *relays* e 1 nó destino. Os nós fontes são responsáveis por gerar pacotes de dados e transmití-los para o nó destino. Os nós *relays* representam nós que não estão gerando dados, e o nó destino somente recebe os pacotes e transmite a mensagem NACK caso necessário. Os nós são dispostos de forma aleatória, sendo reposicionados a cada execução, em uma área quadrada de lado 500 metros a uma profundidade de 70 metros, com exceção do nó destino que é posicionado sempre no centro do quadrado, na coordenada (250,250,70).

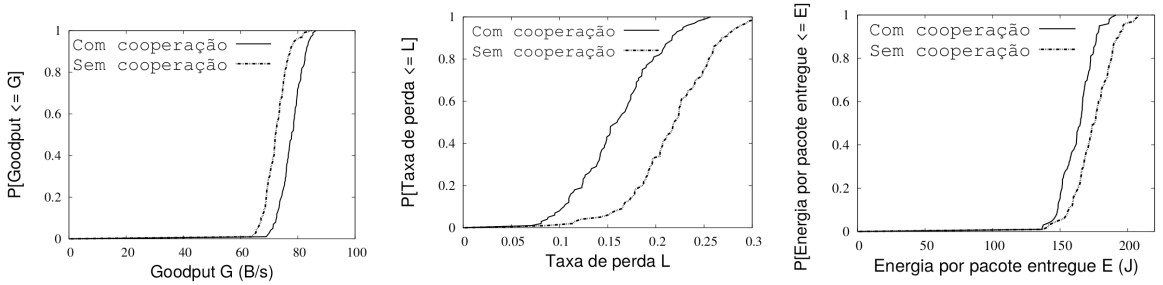
O ns-3 implementa um algoritmo baseado em fluxos e subfluxos. Cada fluxo gera um conjunto de subfluxos que não se sobrepõe [L'ecuyer et al. 2002]. Assim para produzir múltiplas execuções independentes, fixamos o fluxo escolhendo uma valor para a semente e mudamos apenas o subfluxo para cada execução. A semente escolhida foi o valor arbitrário 138 e para cada execução i utiliza-se o subfluxo i.

[1]ns-3 - `https://www.nsnam.org/`, 2017.

Para cada protocolo foram realizadas 100 execuções, e cada execução simula o funcionamento da rede por 10.000 segundos transmitindo pacotes de dados de 800 bytes, e pacotes de controle de 3 bytes. As configurações do transdutor são baseadas no modem acústico WHOI [Freitag et al. 2005]: taxa de dados de 4.000 bps, frequência central de 4.000 Hz e modulação BPSK, consumo de energia para a transmissão de pacotes de 50 W, consumo de energia para recepção de pacotes de 158 mW e consumo de energia em modo ocioso de 158 mW. Já para a configuração dos protocolos, os *slots* possuem duração de 2 s, sendo 1,5 s para a transmissão de dados e 0,5 s como tempo de guarda para evitar a colisão de pacotes. No protocolo proposto os *slots* de controle duram 0,4 s.

4.2. Resultados Numéricos

Os resultados foram avaliados segundo um conjunto de três métricas. A primeira é o *goodput* que corresponde ao total de bytes dos pacotes de dados recebidos pelo nó destino, em relação ao tempo total da simulação. A segunda é a taxa de perda de pacotes e a terceira é a energia total gasta na simulação por pacote entregue.



(a) *Goodput* (b) Taxa de perda de pacote (c) Energia por pacote entregue

Figura 3. Gráficos das funções de distribuição acumulada dos resultados

A Figura 3-a apresenta o *goodput* que, com cooperação, teve média de 77,816 bps e sem cooperação média de 72,476 bps, um ganho médio de 7,36%. A Figura 3-b mostra a taxa de perda de pacote que, com cooperação, teve média de 16,06% e sem cooperação média de 21,82%, um ganho médio de 26,39%. A Figura 3-c mostra a energia total gasta na simulação por pacote entregue com sucesso, tendo média com cooperação de 163,19918 J e sem cooperação de 175,14045 J, um ganho médio de 6,818%

O esquema de cooperação proposto acrescenta ao protocolo um *overhead* devido a mensagens de cooperação WTC e aos períodos de sinalização. Porém, os resultados demonstram que os ganhos obtidos com a cooperação são significativos, especialmente segundo a métrica de energia, conseguindo reduzir o consumo por pacote entregue.

5. Conclusões

Nesse artigo, apresentamos um protocolo MAC cooperativo para redes de sensores aquáticos. Nosso esquema de cooperação se baseia na retransmissão de mensagens que falharam, por nós que estariam ociosos. Os resultados das simulações mostram uma melhoria, principalmente, na taxa de perda de pacotes, que em média, foi reduzida em 26%. A energia gasta por pacote entregue foi reduzida, em média, em 6,81%, indicando que o esquema de cooperação é eficiente em relação ao consumo energético. Trabalhos futuros

devem avaliar a escalabilidade do esquema proposto bem como a cooperação utilizando outros protocolos MAC, além de implementá-los em uma rede real e confrontar os resultados com os alcançados até o momento.

Agradecimentos

Os autores agradecem o apoio de CNPq, CAPES e da FAPEMIG.

Referências

Brekhovskikh, L. M. (2003). *Fundamentals of ocean acoustics*. Springer Science & Business Media.

Carbonelli, C. and Mitra, U. (2006). Cooperative multihop communication for underwater acoustic networks. In *Proceedings of the 1st ACM international workshop on Underwater networks*, pages 97–100. ACM.

Cheng, X., Cao, R., Qu, F., and Yang, L. (2012). Relay-aided cooperative underwater acoustic communications: Selective relaying. In *IEEE OCEANS*, pages 1–7.

Dianati, M., Ling, X., Naik, K., and Shen, X. (2006). A node-cooperative arq scheme for wireless ad hoc networks. *IEEE Trans. on Vehicular Technology*, 55(3):1032–1044.

Domingo, M. C. (2011). A distributed energy-aware routing protocol for underwater wireless sensor networks. *Wireless Personal Communications*, 57(4):607–627.

Felemban, E., Shaikh, F. K., Qureshi, U. M., Sheikh, A. A., and Qaisar, S. B. (2015). Underwater sensor network applications: A comprehensive survey. *International Journal of Distributed Sensor Networks*, 2015:5.

Freitag, L., Grund, M., Singh, S., Partan, J., Koski, P., and Ball, K. (2005). The whoi micro-modem: An acoustic communications and navigation system for multiple platforms. In *OCEANS, 2005. Proceedings of MTS/IEEE*, pages 1086–1092. IEEE.

Heidemann, J., Stojanovic, M., and Zorzi, M. (2012). Underwater sensor networks: applications, advances and challenges. *Phil. Trans. R. Soc. A*, 370(1958):158–175.

L'ecuyer, P., Simard, R., Chen, E. J., and Kelton, W. D. (2002). An object-oriented random-number package with many long streams and substreams. *Operations research*, 50(6):1073–1075.

Lee, J. W., Cheon, J. Y., and Cho, H.-S. (2010). A cooperative arq scheme in underwater acoustic sensor networks. In *OCEANS 2010 IEEE-Sydney*, pages 1–5. IEEE.

Rappaport, T. S. (2002). Wireless communications–principles and practice, (the book end). *Microwave Journal*, 45(12):128–129.

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.

Wang, H., Wang, S., Zhang, E., and Zou, J. (2016). A network coding based hybrid arq protocol for underwater acoustic sensor networks. *Sensors*, 16(9):1444.

Wenz, G. M. (1962). Acoustic ambient noise in the ocean: Spectra and sources. *The Journal of the Acoustical Society of America*, 34(12):1936–1956.

FEED: a Forecast Evaluation of Elasticity cloud Data

Maurício M. Neto[1], **Emanuel F. Coutinho**[1,2], **Gustavo A. C. Santos**[1], **Leonardo O. Moreira**[1,2]

[1]IBITURUNA – Research Group of Cloud Computing and Systems
[2]Virtual University Institute (UFC Virtual) – Fortaleza, CE
Federal University of Ceará (UFC) – Fortaleza, CE – Brazil

{maumneto,gstcampos}@gmail.com, {emanuel,leoomoreira}@virtual.ufc.br

Abstract. *Cloud computing has stood out due to their several characteristics, especially providing self-service on demand to users. For this, the cloud must have elasticity properties to be able to provide a certain service satisfactorily, i.e., the resources must be made available without loss of Quality of Service (QoS). This article presents FEED, a forecast evaluation of elasticity cloud data. The FEED analyzes the validation of a prediction models that best suits the characteristics of the cloud, with the objective of providing the cloud with the ability to allocate the necessary resources to the user in advance, avoiding the breakdown of QoS.*

1. Introduction

Currently, cloud computing has received great attention, highlighting its essential characteristics: self-service on demand, broad access, pooling of resources, rapid elasticity and measured service. With an increase in cloud services availability and their easy accessibility, it is natural that the number of users and applied workloads also grow [Coutinho et al. 2016b]. Consequently, cloud providers should adjust their resources without loss of QoS (Quality of Service) or violation of SLA (Service Level Agreement). In this context, some cloud computational concepts are also emerging, such as elasticity [Coutinho et al. 2016a]. Elasticity is a feature of cloud computing widely discussed both in academic and commercial environments. According to the National Institute of Standards and Technology (NIST) definition, resources can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand [Mell e Grance 2011].

The reactive approaches detect and react to existing resource bottlenecks through defined boundaries of a system or application [Santos et al. 2013]. In contrast, predictive approaches are used to avoid the wearing down a determined computational resource [Santos et al. 2013, Moreira Neto et al. 2016]. These algorithms provide proactive solutions to computational systems, resulting in resource allocations previous the expected need [Santos et al. 2013, Moreira Neto et al. 2016]. Among the various models of prediction, it is possible to mention the large sets of models: the time series and the machine learning models [Santos et al. 2013]. We used in our work the ARIMA and SVR models that represents, consecutively, a time series and a machine learning model.

In this paper, we propose FEED, a **F**orecast **E**valuation of **E**lasticity cloud **D**ata. The FEED approach aims to apply predictive models in a cloud computing environment.

Through this approach, we intend to find a predictive method that better fit to apply in a cloud computing scenario.

2. Background

According to the National Institute of Standards and Technology (NIST) definition, cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources, such as networks, servers, storages, applications, and services, which can be rapidly provisioned and released with minimal management effort or service provider interaction [Mell e Grance 2011]. Elasticity can be defined as the degree to which a system is able to adapt to changes in workloads by resources provisioning and unprovisioning in an autonomic manner, so that at each point in time the available resources combine with the workload demand as close as possible [Herbst et al. 2013]. Therefore, elasticity is becoming a growing need due to the dynamic nature of different applications and workloads.

Coutinho et al. (2016) proposed an architecture for provisioning cloud computing elasticity [Coutinho et al. 2016b], based on some autonomic computing concepts, described by Kephart and Chess [Kephart e Chess 2003], such as: control loops, collectors, actuators, rules, and self configuration. Figure 1 shows the proposed architecture with its components and relationships. In this architecture, workloads are generated by different sources, such as human users, benchmarks, and applications. This workload depending on its goals can be applied only to the load balancer, or to other virtual machines in the infrastructure, creating competition for resources. The requests received by the load balancer are distributed to the allocated virtual machines. Locally, in each virtual machine, collectors store data in the database as defined in local control loop. The global control loop manages the entire flow of the architecture, triggering prediction and running services when needed, and it is also responsible for monitoring whether the consolidated values are exceeding any of the thresholds defined by the rules, and thus, trigger a predefined action to adjust the allocation of resources.

In the scope of this work, we will focus on the prediction component. The prediction component is a mechanism based on statistical techniques (e.g. regression and machine learning techniques), aiming to predict increasing or decreasing demands, for avoiding SLA breaks and idleness. Santos et al. (2013) presented an approach to support dynamic provisioning of computing cloud resources to meet SLA. The presented approach used two prediction models: Autoregressive Integrated Moving Average (ARIMA), a statistical model widely used in time series analysis and Support Vector Machine for Regression (SVR), a model based on machine learning. The effectiveness of the ARIMA and SVR models were validated in reactive and proactive scenarios using different workloads. According to the results, the ARIMA model obtained better results in short-term forecasting scenarios. On the other hand, the SVR had better results in predictions with large data volumes, that is, long-term forecasts.

Moreira et al. (2014) presented an approach to improve quality of service for multi-tenant databases. This approach used the combination of ARIMA and SVR prediction models to predict possible SLA violations in the clouds. For the evaluation, several databases were used, with different sizes, different data schemas and workloads in different situations. The experimental results showed that the ARIMA ob-

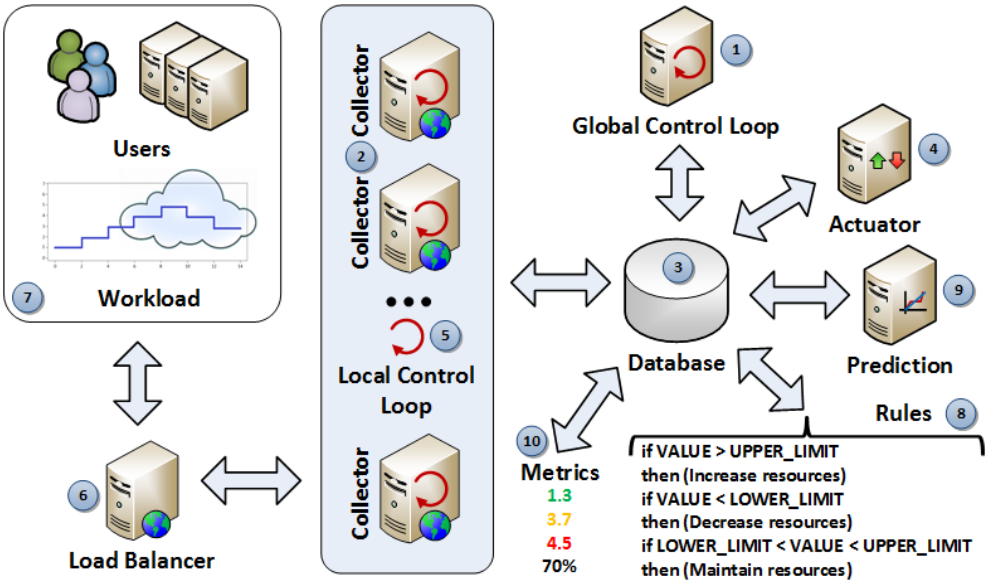


Figure 1. Generic autonomic architecture for elasticity in cloud computing [Coutinho et al. 2016b]

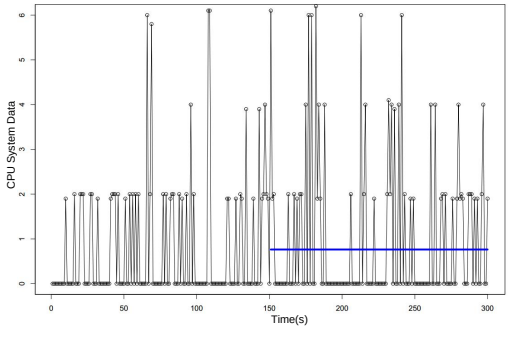
tained the prediction closest to the real situation. In the experiments performed by [Santos et al. 2013] and [Moreira et al. 2014] it can be seen that ARIMA obtained better results. In [Santos et al. 2013] and [Moreira et al. 2014] the standard parameters were used for both models, ARIMA and SVR. The SVR requires an effort in the parameterization step. ARIMA is simpler to parameterize, just enter the time series and the prediction window. The ARIMA is automatically adjusted to each observation window, adapting it to each prediction scenario.

3. Experiments and Discussion

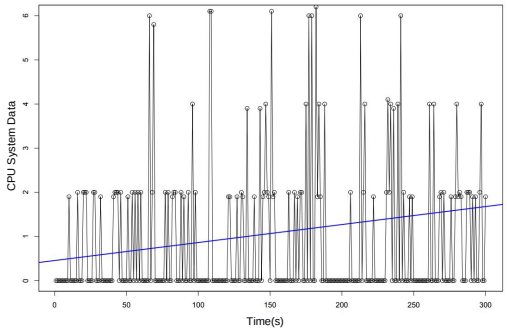
We implemented FEED in R language to generate the predictive analyses with the elasticity database provided by [Coutinho et al. 2016c]. We analyzed three prediction methods: ARIMA, SVR and Linear Regression. A set of experiments were performed to evaluate the predictive methods impacts in a cloud elasticity database. Among the various parameters that can be used to characterize elasticity in cloud, we used the CPU rate provided by a stress test in the Azure cloud. The CPU usage rate was chosen because it is a factor directly related to the cloud applications. The CPU usage rate ranged from 0 to 60% over time. The experiments aim to mitigate which method of prediction best fits to predict the elasticity data in the cloud. For this, the proposed experiments vary both the windows of data inputs and the prediction windows. The values for the prediction windows were chosen empirically. Experiment I aim to evaluate the prediction methods in a cloud elasticity data base with a 50% input window to predict 50%. Experiment II was similar to Experiment I, however, using the input window with 70% and predicting the remaining 30%. All experiments were performed in a machine with processor core i7 2.5GHz and 8GB RAM memory. We performed analyzes of the predictive methods in each experiment and analyzes of the prediction errors such as Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). Some statistical analyzes were made (e.g. standard deviation and variance of data obtained). The simulation time was also analyzed for each method.

3.1. Experiment I

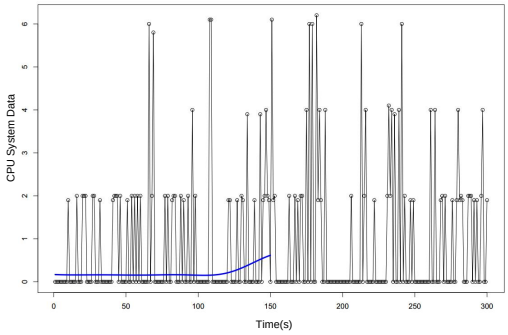
Experiment I aims to evaluate the predictive methods in a mid-term prediction window. Figure 2(a), 2(b) and 2(c) show the graphs of the predictive methods taking as input 50% of the CPU system data. Figure 2(a) shows ARIMA method tends to average the input values. Due to the irregularity of the input data, the ARIMA failed to obtain a good prediction of the data. However, the linear regression obtained a better result than the other methods analyzed, as pointed in Figure 2(b). The SVR method obtained a poor performance in relation to the other methods in this experiment.



(a) ARIMA



(b) Linear Regression



(c) SVR

Figure 2. Experiment I: Analysis with 50% of input data.

Table 1. Analyzes of the experiment I.

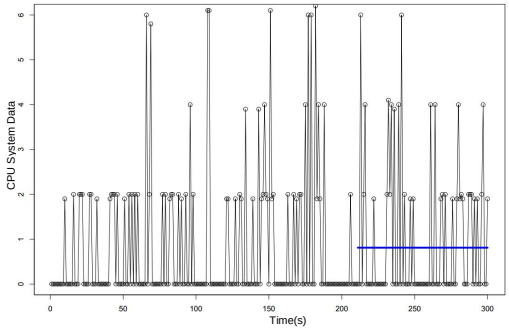
	RMSE	MAE	Variance	SD	ST (ms)
ARIMA	1.627	1.233	3.805e-33	6.168e-17	0.28
SVR	1.465	0.813	0.0008	0.029	0.33
Linear Regression	1.319	1.041	0.0313	0.177	0.1

Legend: SD = Standard Deviation, ST = Simulation Time

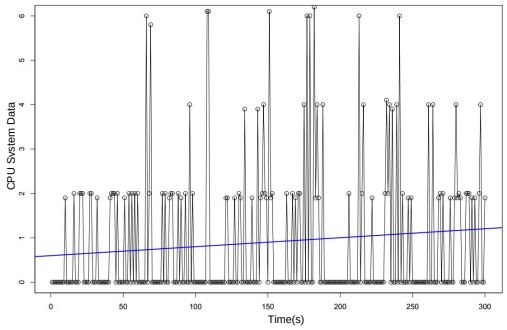
Table 1 shows ARIMA had the higher prediction error among the other methods. It is also possible to observe that the variance and standard deviation values of the ARIMA were low due to its tendency to mean the input values. The SVR has obtained result close to ARIMA but also tended to average the data, not presenting a good result. Linear regression obtained the best behavior among the other prediction methods (Figure 2(b)). Table 1 shows the simulation times of the methods ranged from 0.1 to 0.33 milliseconds. The linear regression achieved the shortest simulation time due to its low complexity.

3.2. Experiment II

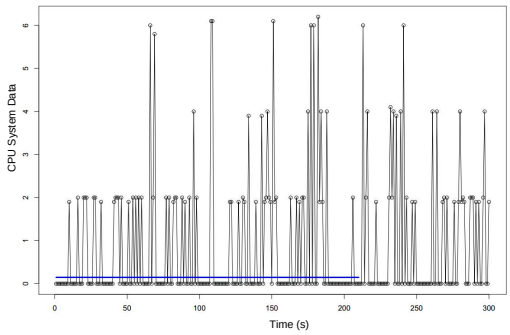
Experiment II aims to evaluate the predictive methods in a long-term prediction window. Figure 3(a), 3(b) and 3(c) show the graphs of the predictive methods taking as input 70% of the CPU system data. Figure 3(a) and 3(c) shows ARIMA and SVR maintained the average data behavior, i.e., not being successful in predicting the data provided. The linear regression also showed a behavior similar presented in Experiment II (Figure 3(b)).



(a) ARIMA



(b) Linear Regression



(c) SVR

Figure 3. Experiment II: Analysis with 70% of input data.

Table 2. Analyzes of the experiment II.

	RMSE	MAE	Variance	SD	ST (ms)
ARIMA	1.558	1.239	7.48e-33	8.65e-17	0.30
SVR	1.691	1.067	0.0006	0.033	0.28
Linear Regression	1.457	1.134	0.015	0123	0.16

Legend: SD = Standard Deviation, ST = Simulation Time

Table 2 shows SVR model obtained the highest prediction error values again and this occurs due to the SVR tendency to average of input data. ARIMA maintained the results pattern. Although the SVR is a robust method, the database is inconsistent, making it difficult to predict the method. The linear regression obtained the best result among the other methods in Experiment II, however, the error results in this experiment were worse than in Experiment I. The linear regression show a better fit in mid-term prediction window in comparison a long-term window. The simulation time of this experiment also range from 0.1 to 0.3 milliseconds. The linear regression had a longer simulation time compared to the Experiment I, this was caused by the higher number of input data.

4. Conclusion and Future work

This paper presented FEED, a forecast evaluation of elasticity cloud data. The FEED's purpose is the evaluation of a predictive model that best suits elasticity demands in the cloud. Three prediction methods were evaluated in our experiments (ARIMA, SVR and Linear Regression) and the CPU system data was used as input to these prediction models. It were performed experiments to evaluate the models for different sort of prediction windows (e.g., mid- long-term). The results of the experiments show that the linear regression model had a better performance than the other models compared. As future work, we intend to analyze other models (e.g., neural networks and genetic algorithms) and observe the behavior of the models in short-term prediction windows. Due to the varied behavior of the data base, we intend to use other cloud-related metrics as input to the predictive methods and also uses outliers detection and smoothing techniques.

References

Coutinho, E. F., Rego, P. A. L., e de Souza, J. N. (2016a). Evaluating the elasticity of multimedia applications in a cloud computing environment using network metrics. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, pages 455–461, New York, NY, USA. ACM.

Coutinho, E. F., Rego, P. A. L., Gomes, D. G., e de Souza, J. N. (2016b). An architecture for providing elasticity based on autonomic computing concepts. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, pages 412–419, New York, NY, USA. ACM.

Coutinho, E. F., Rego, P. A. L., Gomes, D. G., e de Souza, J. N. (2016c). Physics and microeconomics-based metrics for evaluating cloud computing elasticity. *Journal of Network and Computer Applications*, 63:159 – 172.

Herbst, N. R., Kounev, S., e Reussner, R. (2013). Elasticity in cloud computing: What it is, and what it is not. In *Proceedings of the 10th International Conference on Autonomic Computing(ICAC 2013), San Jose, CA*, pages 23–27. USENIX.

Kephart, J. O. e Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36(1):41–50.

Mell, P. M. e Grance, T. (2011). Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, United States.

Moreira, L. O., Farias, V. A. E., Sousa, F. R. C., Santos, G. A. C., Maia, J. G. R., e Machado, J. C. (2014). Towards improvements on the quality of service for multi-tenant rdbms in the cloud. In *Data Engineering Workshops (ICDEW), 2014 IEEE 30th International Conference on*, pages 162–169.

Moreira Neto, M., Moreira, L. O., e Gomes, D. G. (2016). FLECHA: a Forecasting eLEction meCHAnism for semantic collectors sensor nodes. pages 2851 – 2862.

Santos, G. A. C., Maia, J. G. R., Moreira, L. O., Sousa, F. R. C., e Machado, J. C. (2013). Scale-space filtering for workload analysis and forecast. In *2013 IEEE Sixth International Conference on Cloud Computing*, pages 677–684.

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