Gerenciamento de Buffer Baseado em Egoísmo para Redes Tolerantes a Atrasos e Desconexões

Camilo B. Souza[1], **Edjair Mota**[1], **Leandro Galvão**[1] **Diogo Soares**[1]

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Avenida General Rodrigo Octávio Jordão Ramos – 3000 – Manaus – AM – Brasil

{camilo.souza,edjair,galvao,diogo.soares}@icomp.ufam.edu.br

Abstract. *The delivery rate in delay tolerant networks is directly impacted by the buffer management algorithm. Recently, by considering the social characteristics of the network node brought a new point of view to the design of such algorithms. This work aims to propose an algorithm that takes into account the selfishness of each node when it has to discard a message from its buffer. A learning machine technique estimates the friendship strength among the nodes. To improve this classification, the proposed algorithm makes use of a training database from an experiment carried out with users from the MIT campus. By means of stochastic simulation, the validation tasks used the Reality and Cambridge traces for the sake of comparison with similar works. The results showed that the proposed algorithm increased the delivery rate while reducing both the average delivery time and the number of hops to deliver the message to the destiny.*

Resumo. *Em redes tolerantes a atrasos e desconexões, a taxa de entrega é diretamente influenciada pelo algoritmo de gerenciamento de buffer. Recentemente, a consideração de características sociais trouxe um novo ângulo de visão no projeto desses algoritmos. O algoritmo apresentado neste trabalho leva em consideração o egoísmo dos nós para decidir que mensagem descartar do buffer. A força da amizade entre dois nós, preponderante para essa decisão, é classificada utilizando-se uma técnica de aprendizagem de máquina. Para melhorar a qualidade dessa classificação, utiliza-se como dados de treinamento características de amizades extraídas de um experimento realizado por pesquisadores do MIT com usuários do campus universitário. Utilizou-se simulação estocástica baseada nos traces Reality e Cambridge para efeito de comparação com resultados de trabalhos similares. Os resultados obtidos mostram que o algoritmo proposto contribui para o aumento da taxa de entrega na rede, diminuição do tempo médio de entrega, além de uma razoável média de saltos para que uma mensagem alcance o seu destinatário.*

1. Introdução

Redes tolerantes a atrasos e desconexões (DTN) compartilham de características comuns tais como longos atrasos e frequentes desconexões causados principalmente pelo mau funcionamento dos protocolos que atuam na pilha TCP/IP em decorrência da ausência de uma ou mais premissas básicas necessárias para o seu bom funcionamento [de Oliveira et al. 2007].

A camada de agregação proposta por Fall et al. [Fall 2003], posicionada entre a camada de aplicação e a camada de transporte do referido modelo, permite a implementação do paradigma armazena-carrega-encaminha. Na ausência de um caminho fim-a-fim entre o nó origem e nó destino, é possível armazenar de forma persistente uma mensagem [de Oliveira et al. 2007]. No entanto, a combinacão desse paradigma de rede com algoritmos de roteamento baseados em replicações de mensagens, pode causar a situação denominada de *overflow* [Naves et al. 2012], e uma ou mais mensagens devem ser escolhidas para serem descartadas do *buffer* dos nós .

Algoritmos de gerenciamento de *buffer* para DTN auxiliam na decisão de que mensagem(ns) retirar em casos de *overflow* do *buffer* dos nós. Na literatura encontram-se diversas propostas de algoritmos de gerenciamento de *buffer* para DTN. Tang et al. [Tang et al. 2012] propuseram uma classificação para os algoritmos de gerenciamento de *buffer* existentes dividindo-os em duas categorias: algoritmos que não usam informações da rede e algoritmos que usam informações da rede. Recentemente, a consideração de características sociais forneceu um novo ângulo de visão no projeto de algoritmos para DTN [Xia et al. 2015], surgindo, assim, uma terceira categoria de algoritmos de gerenciamento de *buffer* baseada em características sociais. Dentre as características sociais mais abordadas estão o egoísmo, a amizade, a centralidade, a popularidade, dentre outras.

Este trabalho apresenta o projeto e a avaliação de desempenho de um algoritmo de gerenciamento de *buffer* para DTN, aqui denominado *Selfish Drop-Based* (SDB), que leva em consideração o egoísmo dos nós que compõem a rede. Para isso, utilizou-se a classificação do egoísmo dos nós realizada por [Zhu et al. 2013], a qual divide o egoísmo de um nó em social ou individual. O nó egoísta individual é aquele que não aceita mensagens destinadas a outros nós; está interessado somente na sua participação dentro da rede. O nó egoísta social aceita mensagens destinadas a outros nós, porém somente àqueles com quem ele tem alguma relação social.

Para realizar o gerenciamento do *buffer* dos nós, o algoritmo SDB classifica os nós quanto ao seu nível de egoísmo e escolhe que mensagem deve ser descartada. Se o egoísmo do nó for classificado como individual, o nó descarta a mensagem candidata a entrar no *buffer*, se for classificado como social, o nó prioriza aquelas destinadas a outros nós com quem ele tem uma amizade forte. Dependendo das limitações de seus recursos (nível de bateria, por exemplo), o nó pode tornar-se egoísta. Nó não egoísta e sem limitações de recursos, executa o descarte levando em consideração a força da amizade entre os nós.

As principais contribuições deste trabalho são:

- considerar o egoísmo no projeto de algoritmos para o gerenciamento do *buffer* dos nós em DTN – uma vez que DTN podem ser formadas por humanos equipados com dispositivos móveis, e o seu comportamento influencia na maneira como são gerenciados os recursos dos dispositivos que compõem a rede, é de suma importância levar em consideração o egoísmo no projeto de algoritmos de gerenciamento de *buffer* para DTN. O algoritmo proposto nesse trabalho visa preencher essa lacuna observada na literatura.
- considerar que nós não egoístas podem se tornar egoístas por conta da criticidade dos seus recursos – os algoritmos para DTN que consideram o egoísmo dos nós, consideram que os nós não egoístas não serão egoístas em nenhuma situação.

Nesse trabalho, considera-se que os nós não egoístas podem tornar-sem egoístas individuais em situações de criticidade dos seus recursos, para garantir o uso do seu dispositivo em favor de si mesmo.

- classificar a força da amizade entre os nós usando técnicas de aprendizagem de máquina – os algoritmos para DTN que consideram a amizade entre os nós, utilizam determinadas métricas para a definição da força da amizade entre os nós. O algoritmo proposto no presente trabalho utiliza uma abordagem diferenciada para classificar a força da amizade entre os nós, a qual inclui o uso de um algoritmo de aprendizagem de máquina combinado com dados de amizades extraídas de um experimento realizado por pesquisadores do MIT com usuários do campus universitário.

O restante desse trabalho está organizado da seguinte forma: na seção 2 são apresentados os principais trabalhos relacionados ao tema abordado nesse trabalho, enquanto que na seção 3 maiores detalhes sobre o algoritmo proposto são dados. A seção 4 apresenta detalhes sobre a metodologia de avaliação, enquanto que a seção 5 aborda os resultados obtidos nas avaliações realizadas. Finalmente, na seção 6 apresentam-se as conclusões e os trabalhos futuros.

2. Trabalhos relacionados

Em DTN, a taxa de entrega é diretamente influenciada pelo algoritmo de gerenciamento de *buffer* [Naves et al. 2012], visto que a decisão de descartar ou não uma determinada mensagem pode possibilitar ou impossibilitar sua entrega em contatos posteriores. Os algoritmos que não utilizam informações da rede, geralmente realizam apenas uma escolha simples para tomar a decisão de descarte. Esses algoritmos utilizam informações das mensagens como por exemplo o tempo de vida da mensagem (TTL) e o tamanho da mensagem. Exemplos de algoritmos pertencentes a esta classe são as seguintes: descarte aleatório [Rashid et al. 2011], descarte da mensagem menos recentemente recebida [Lindgren and Phanse 2006], descarte da mensagem mais antiga no *buffer* [Rashid et al. 2011], descarte da última mensagem na fila [Lindgren and Phanse 2006], descarte da primeira mensagem na fila [Rashid et al. 2011], descarte da mensagem mais nova [Rashid et al. 2011] e descarte da mensagem de maior tamanho [Lindgren and Phanse 2006].

Os algoritmos que utilizam informações da rede, por sua vez, utilizam uma abordagem mais sofisticada levando em consideração informações como o histórico de encontros dos nós da rede [Krifa et al. 2008], a frequência de encontros entre os nós da rede [Tang et al. 2012], a probabilidade de entrega e a estimativa de disseminação de uma mensagem na rede [Naves et al. 2012]. No entanto, mais recentemente a consideração de características sociais trouxe um novo ângulo de visão no projeto de algoritmos para DTN e uma terceira classe de algoritmos de gerenciamento de *buffer* surgiu. Esses algoritmos fazem uso em suas estratégias de características como amizade, egoísmo, centralidade, popularidade, similaridade, entre outras.

Em [Souza et al. 2014], Souza et al. propõem um algoritmo denominado *Drop Less Known* (DLK) para o gerenciamento de *buffer* dos nós de redes DTN. Esse algoritmo introduz uma métrica denominada força da relação social, a qual considera o quão forte são os laços sociais entre dois nós (forte, médio, fraco). Para classificar a relação

social entre um par de nós, os autores utilizaram a quantidade de contatos e um log de ligações entre os nós disponibilizados em um conjunto de dados denominado *Reality*. Os experimentos realizados mostraram que o algoritmo DLK contribuiu para um aumento na taxa de entrega e houve diminuição razoável do atraso médio de entrega de mensagens na rede.

No presente trabalho, os dados do trace *Reality* são usados como dados de treinamento para que um algoritmo de aprendizagem de máquina classifique a força da amizade entre dois nós (forte, fraco). Uma outra diferença deste trabalho para [Souza et al. 2014] está na forma como é feita a classificação da relação social, visto que o algoritmo DLK atribui de maneira estática (antes da simulação iniciar) a força da relação social e essa relação não muda, enquanto que a classificação da relação social (amizade) feita no presente trabalho é feita de maneira dinâmica e por esse motivo pode se alterar durante a simulação, representando assim um comportamento mais realista.

Uma outra característica social abordada em algoritmos de gerenciamento de *buffer* para DTN é a centralidade, que pode ser definida como a quantidade de interação de um nó com os demais nós da rede. Quanto maior for a centralidade de um nó, pode-se concluir que maior interação com outros nós ele tem. Dessa forma, [Settawatcharawanit et al. 2013] propõem um algoritmo para o gerenciamento de *buffer* baseado na centralidade dos nós dentro de comunidades locais. Para calcular a centralidade de um nó dentro de uma comunidade local, o algoritmo analisa as médias de encontros entre os nós em determinadas janelas de tempo. Quando ocorre o *overflow*, o nó procura em seu *buffer* uma mensagem destinada a um nó que não pertença a mesma comunidade que a sua. Se todas as mensagens armazenadas no *buffer* são destinadas a nós da mesma comunidade, a mensagem mais antiga é retirada do *buffer*. Assim como esses autores, no presente trabalho também considera-se a quantidade de encontros para determinar a relação social (amizade), no entanto, consideram-se outras informações que contribuem para aumentar a qualidade da classificação da relação social realizada, a saber: histórico de trocas de mensagens de texto e ligações e a duração dos contatos.

Apesar de haver uma quantidade significativa de trabalhos e de consenso sobre a arquitetura geral das redes DTN [Cerf et al. 2007], ainda não existe um consenso sobre algoritmos de gerência de *buffer* para as mesmas. De acordo com [Tang et al. 2012], os algoritmos encontrados na literatura podem melhorar o desempenho da rede em um determinado grau, mas todos têm suas limitações. Nesse intuito, este trabalho apresenta uma política de gerência de *buffer* que utiliza uma característica social ainda não abordada nesse tema, trata-se do egoísmo entre os nós. A seção a seguir apresenta maiores detalhes sobre o algoritmo proposto.

3. Selfish Drop-Based Algorithm

O algoritmo proposto no presente trabalho leva em consideração o egoísmo dos nós para tomar a decisão de que mensagem descartar do *buffer*. As principais motivações para se gerenciar o *buffer* dos nós utilizando essa característica como base são:

(i) tornar a decisão do descarte de mensagens mais realista - de acordo com [Li et al. 2010], a maioria dos algoritmos em DTN considera que os usuários estão dispostos a carregar e encaminhar mensagens uns para os outros. No entanto,

no mundo real as pessoas são socialmente egoístas e tendem a carregar mensagens somente para aqueles usuários com quem eles tem alguma relação social, e a vontade de carregar uma mensagem destinada a outra pessoa depende da força da relação social entre eles. Dessa forma, como no gerenciamento de *buffer* uma ou mais mensagens devem ser descartadas em caso de *overflow*, é razoável supor que usuários reais, se questionados sobre que mensagens descartar do *buffer* do seu dispositivo, escolheriam aquelas destinadas a usuários com quem a relação social entre eles é mais fraca ou inexistente. Assim, o algoritmo SDB é uma interessante alternativa para atender esse fato.

(ii) aumentar a taxa de entrega através do egoísmo social - de acordo com [Zhu et al. 2013], o egoísmo social pode ser definido como o tipo de egoísmo em que um nó aceita participar das ações da rede encaminhando mensagens para outros nós, no entanto só aceita carregar mensagens destinadas a outros nós com quem tem certa relação social. Baseado nesse fato, chamando-se de círculo social o grupo de nós com quem se tem certa relação social, é razoável supor que os nós que pertencem ao círculo social de um determinado nó, são também aqueles com quem ele se encontra com uma determinada frequência. Como em DTN a oportunidade de contato é de suma importância para que dois nós se comuniquem, pode-se concluir que quanto maior for a frequência de contatos entre dois nós, maior será a quantidade de dados trocados por eles. Dessa forma, como as oportunidades de contatos são maiores, consequentemente a taxa de entrega de mensagens tende a aumentar.

O algoritmo 1 apresenta o pseudocódigo do funcionamento geral do algoritmo SDB. Basicamente, em uma oportunidade de contato, o algoritmo SDB inicia sua operação verificando se o nó candidato a receber mensagens é egoísta ou não egoísta, e, dependendo dessa verificação incial o algoritmo irá executar um conjunto de operações diferentes. Para melhor entendimento do leitor, as explicações sobre o funcionamento do algoritmo SDB foram divididas em duas subseções, uma que aborda a operação do algoritmo SDB caso o nó candidato a receber a mensagem seja egoísta (chamada de Procedimento 1) e outra que aborda a situação em que o nó candidato é não egoísta (chamada de Procedimento 2).

Algoritmo 1 *Selfish Drop-Based Algorithm*

```
1: procedure SDB(noRecebedor, buffer[])
2:     noRecebedor.tipo ← determinaTipo()
3:     if noRecebedor.tipo ==“egoista” then procedimento1()
4:     elseprocedimento2()
5:     end if
6: end procedure
```

3.1. Procedimento 1: candidato a receber a mensagem é egoísta

O pseudocódigo 2 apresenta maiores detalhes sobre o funcionamento do algoritmo SDB caso o nó seja identificado como egoísta. Se o nó candidato a receber a mensagem for egoísta, o algoritmo SDB continua as verificações analisando se o tipo de egoísmo do nó candidato a receber a mensagem é individual ou social. Caso seja individual, o algoritmo

SDB descarta a mensagem candidata a entrar no *buffer*. Caso seja social, o algoritmo SDB descarta a mensagem destinada ao nó com quem se tem a relação social mais fraca. Como dito anteriormente, a relação social escolhida para auxiliar na tomada de decisão pelo algoritmo SDB foi a amizade entre os nós, que, no presente trabalho pode ser classificada como fraca ou forte. É interessante destacar ainda que para fazer a classificação da força da amizade, o algoritmo SDB utiliza uma técnica de aprendizagem de máquina denominada Naive Bayes e uma base de dados retirados de um experimento real denominado *Reality* [Eagle and Pentland 2006]. Maiores detalhes sobre essa funcionalidade são dadas a seguir.

3.1.1. Classificação da amizade

Para classificar a força da amizade entre um par de nós na rede, o algoritmo SDB utiliza um algoritmo de aprendizagem de máquina denominado Naive Bayes [Gama and teorema de Bayes 2012, Lewis 1998]. Para escolher-se o algoritmo de aprendizagem de máquina observou-se em outros trabalhos da literatura o desempenho dos principais algoritmos testados, detectando-se que em vários deles o algoritmo escolhido obteve o melhor desempenho na tarefa de classificação [Gama and teorema de Bayes 2012, Lewis 1998].

O algoritmo Naive Bayes é considerado um classificador supervisionado, ou seja, utiliza instâncias juntamente com seus rótulos para classificar novas instâncias. Nesse contexto, uma instância é um exemplo conhecido de valores para os atributos ou características que ajudam na classificação dos rótulos, os quais se tratam de um valor para a classe que se deseja classificar. Por exemplo, supondo que os atributos de interesse no problema em questão sejam o *Round Trip Time*(RTT), o tamanho da fila dos roteadores e a largura de banda disponível. Considerando ainda que os valores de RTT variem entre 50 a 200ms, o tamanho da fila de 0 a 100 , a largura de banda disponível de 0 a 100 e que no problema existam dois rótulos (classes) possíveis : 1 (presença de congestionamento) e 2 (ausência de congestionamento). Nesse contexto, a tupla de valores (75, 50, 80, presença de congestionamento) é considerada um exemplo já conhecido (instância) dentro do problema. De maneira geral, em seu funcionamento, o algoritmo Naive-Bayes, basicamente analisa os valores para os atributos de um exemplo desconhecido e tenta predizer a que classe estes pertencem, baseando-se no histórico de probabilidades para exemplos conhecidos (base de treino).

Algoritmo 2 *Procedimento 1*

```
1: procedure PROCEDIMENTO1(noRecebedor, buffer[], M1)
2:     noRecebedor.tipoEgoismo ← determinaTipo()
3:     if noRecebedor.tipoEgoismo ==("individual") then descartarM1
4:     elseverificarAmizade()
5:     end if
6: end procedure
```

O algoritmo Naive Bayes é baseado no Teorema de Bayes, que é utilizado para o cálculo das probabilidades necessárias para a classificação de uma nova instância. No contexto de aprendizagem de máquina, pode-se definir o Teorema de Bayes da seguinte

forma: dada uma instância desconhecida *A = (a1, a2, a3.. an)*, onde *a1, a2, a3,...an* são valores dos atributos de uma instância, deseja-se predizer sua classe. Para realizar a escolha da classe, o Teorema de Bayes utiliza a seguinte Fórmula:

$$P(Classe|A) = \frac{P(A|Classe) \times P(Classe)}{P(A)} \tag{1}$$

Para cada atributo da instância *A = (a1, a2, a3.. an)*, pode-se reescrever:

$$P(Classe|a1, ..., an) = \frac{P(a1, ..., an|Classe) \times P(Classe)}{P(a1, ..., an)} \tag{2}$$

Em termos gerais, a equação (2) representa a probabilidade de a nova instância ser de uma determinada classe em termos dos valores de cada um dos atributos *A = (a1, a2, a3.. an)*. Além disso, ao observar-se a Equação (2), percebe-se que o denominador é uma constante, sendo assim possível anulá-lo do Teorema de Bayes. Com isso, o Teorema de Bayes fica resumido a:

$$argmax P(a1, ...an|Classe) \times P(Classe) \tag{3}$$

O Teorema de Bayes faz a suposição de que os atributos de uma instância são estatisticamente independentes, ou seja, um determinado valor x para um atributo não implica diretamente em um valor y para outro atributo. No entanto, na maioria das vezes, a suposição de independência entre atributos acaba se tornando falsa. Apesar disso, o algoritmo Naive Bayes produz resultados bastante satisfatórios em diversos problemas reais, ou seja, consegue predizer de maneira correta a classe de uma determinada instância desconhecida [Domingos and Pazzani 1997].

No presente trabalho, o problema para o qual utilizou-se o algoritmo Naive Bayes como classificador pode ser definido da seguinte forma:

- **Tarefa realizada**: classificar a força da amizade entre dois nós em uma rede. As classes possíveis são Amizade Forte ou Amizade Fraca.
- **Experiência de treinamento**: uma base de dados derivada de um experimento em que a força da amizade entre os nós participantes já são conhecidas e definidas como fortes ou fracas.
- **Atributos utilizados como base para a classificação**: quantidade de contatos, duração dos contatos, log de ligações, log de mensagens de texto.

Os atributos utilizados como base para a classificação podem ser descritos da seguinte forma:

- **quantidade de contatos** – o total de encontros entre dois nós em uma semana.
- **duração de contatos** – tempo médio de duração dos encontros em uma semana.
- **log de ligações** – quantidade de ligações trocadas entre dois nós em uma semana.
- **log de mensagens de texto** – quantidade de mensagens de texto trocadas entre dois nós em uma semana.

A base de treino utilizada para classificar novas instâncias foi extraída de um experimento real realizado por pesquisadores do MIT denominado *Reality Mining* [Pentland et al. 2009]. O projeto foi realizado durante seis meses com 97 pessoas equipadas com dispositivos móveis com a interface Bluetooth ativada. Como resultado desse experimento, foi disponibilizado para a comunidade científica diversas informações referentes ao uso dos dispositivos móveis pelas pessoas, tais como conectividade, proximidade, localização e outras atividades dos usuários.

Um dos principais motivos para usar-se essa base de dados no presente trabalho é o log de ligações e de mensagens de texto dos usuários que participaram do experimento. Como citado anteriormente, essa informação é usada como base para classificar a força da amizade entre dois nós na rede. Um outro motivo importante para utilização dessa base de dados no presente trabalho são as respostas para um questionário aplicado aos participantes. Dentre outras perguntas, neste questionário existia uma questão com o seguinte conteúdo: "essa pessoa pertence ao seu ciclo de amizades?". As respostas possíveis eram sim ou não. Dessa forma, para criar-se a base de treino utilizada no presente trabalho, foram extraídas do conjunto de dados *Reality* as seguintes informações sobre a relação entre um par de nós A e B: log de informações e mensagens de texto, quantidade e duração de contatos entre eles e qual a relação entre eles (se são amigos ou não).

3.2. Procedimento 2: candidato a receber a mensagem é não egoísta

O pseudocódigo 3 apresenta maiores detalhes sobre o funcionamento do algoritmo SDB caso o nó seja identificado como não egoísta. Se o nó candidato a receber mensagens é não egoísta, o algoritmo SDB avalia se o mesmo pode se tornar egoísta por conta das limitações dos seus recursos. No presente trabalho, levou-se em consideração o nível de energia como critério para avaliar se um nó não egoísta tornou-se egoísta. A ideia por trás dessa funcionalidade do algoritmo surgiu da hipótese de que em determinadas situações, usuários reais desejariam não participar da rede por conta do nível crítico de recursos do seu dispositivo.

Algoritmo 3 *Procedimento 2*

1: **procedure** PROCEDIMENTO2($noRecebedor$, ***buffer***$[]$, $M1$)
2: $noRecebedor.nivelBateria \leftarrow obtemNivelBateria()$
3: **if** $noRecebedor.nivelBateria <=$"20%" **then** $descartarM1$
4: **else**$verificarAmizade()$
5: **end if**
6: **end procedure**

Essa hipótese foi validada através da execução de um pequeno experimento com objetivo de avaliar duas questões, a saber: (i) como usuários reais se comportariam em situações de participação da rede com nível de energia crítico e (ii) a partir de que nível de energia os usuários considerariam participar da rede somente em favor de si próprio (não aceitando mensagens para os demais). Para responder essas questões, executou-se um experimento através da aplicação de um questionário contendo duas questões com a participação de 100 pessoas. Os participantes do experimento foram pessoas de diversos cursos diferentes da Universidade Federal do Amazonas. Além disso, o questionário permaneceu disponível para preenchimento durante um mês. O conteúdo do questionário é apresentado abaixo:

> "Imagine que você está em um lugar isolado onde a comunicação só possa ser feita por um aplicativo de troca de mensagens chamado MSG1. Porém, diferente de outros aplicativos , no MSG1 para você se comunicar com outra pessoa você precisa "parear"seu dispositivo com o da outra pessoa via Wifi ou Bluetooth enviando antes um convite para se conectar com aquela pessoa. Neste cenário, responda: "

- Pergunta 1 – Se uma pessoa envia uma solicitação de conexão, porém a bateria do seu dispositivo está em um nível crítico, o que você faria? (a) aceitaria o convite (b) não aceitaria o convite (c) esperaria recarregar a bateria para depois comunicar com a pessoa.
- Pergunta 2 – A partir de qual nível você considera que a bateria do seu dispositivo está em um nível crítico? (a) abaixo de 10% (b)de 10% a 20% (b)21% a 30%

A Tabela 1 apresenta os resultados obtidos nesse experimento. Como pode-se observar, 55% das pessoas responderam que "esperariam recarregar a bateria para depois comunicar com a pessoa". Considerando-se que apenas 10% das pessoas respondeu que aceitaria o convite para se comunicar, pode-se concluir que de certa forma, 90% das pessoas deixaria para se comunicar posteriormente. A conclusão que pode ser tirada desse experimento é que nessa situação de criticidade no nível de bateria dos dispositivos dos nós, o nó candidato a receber uma mensagem deve decidir não recebê-la por que deseja economizar energia em favor da sua própria participação na rede. Dessa forma, é razoável supor que nesse caso nós não egoístas tornem-se temporariamente egoístas do tipo individual e portanto devem descartar a mensagen candidata a entrar no *buffer*. Além disso, nesse trabalho considera-se um dispositivo em nível crítico de bateria quando o seu valor está abaixo de 20%, baseando-se nas respostas obtidas no questionário.

Tabela 1. Respostas em % para as perguntas do questionário aplicado no experimento

Pergunta	(a)	(b)	(c)
1	10%	35%	55%
2	28%	67%	5%

Para efeito de simulação, no presente trabalho assumiu-se que 40% dos nós participantes da simulação são egoístas (20 % individual e 20%social), enquanto que 60% dos nós são não egoístas.

4. Metodologia

Para validar a proposta de gerência de *buffer* baseada em egoísmo para redes DTN, utilizou-se a técnica da simulação estocástica. Nesse trabalho, implementou-se um modelo de simulação que inclui o algoritmo *Selfish Drop-Based Algorithm* e outros algoritmos disponíveis na literatura, permitindo assim quantificar o desempenho do algoritmo proposto. Os algoritmos escolhidos para comparação foram os algoritmos *Drop Less Known* [Souza et al. 2014], e o algoritmo proposto por [Settawatcharawanit et al. 2013], doravante chamado de *Drop Centrality-Based* (DCB). A motivação para a escolha desses algoritmos é a similaridade das suas propostas que também são baseadas em características sociais, da mesma forma que o algoritmo proposto no presente trabalho. O cenário

de simulação foi implementado no simulador *Opportunistic Network Environment* (The ONE) [Keränen et al. 2009]. O simulador The ONE foi selecionado levando em consideração os demais trabalhos na literatura que trabalharam com simulação, os quais em sua grande maioria utilizaram o referido simulador para a implementação do seu modelo.

Para simular a mobilidade dos nós, utilizou-se traces de mobilidade real escolhendo-se dois traces bem conhecidos disponíveis na literatura, chamados de *Cambridge* e *Reality*. A escolha de traces de mobilidade real em vez de mobilidade sintética deu-se principalmente pelos inconvenientes citados nesta última forma de mobilidade em [Yoon et al. 2003, Resta and Santi 2002]. A Tabela 2 apresenta maiores detalhes sobre os traces utilizados neste trabalho. Utilizou-se ainda como algoritmo de roteamento o protocolo Epidêmico [Vahdat et al. 2000]. A principal motivação para utilização desse protocolo é a maneira como o mesmo replica as mensagens na rede, favorecendo o *overflow* e consequentemente necessitando de um bom gerenciamento de *buffer*.

Tabela 2. Traces baseados em movimentação humana real

Trace	Dispositivo	Tipo de rede	Número de Nós	Duração	Granularidade	Número de Contatos
Cambridge	*iMotes*	*Bluetooth*	36	11 dias	100 segundos	10873
Reality	*Telefone*	*Bluetooth*	97	246 dias	300 segundos	54667

Segundo [Fathima e Wahidabanu, 2011], o desempenho de uma rede DTN é medido em termos da razão da taxa de entrega e do atraso médio de entrega. Dessa forma, realizou-se uma comparação do desempenho de várias políticas de gerência de *buffer* avaliando-se três métricas: taxa de entrega, atraso médio e média de saltos.

Para simular o consumo de energia dos nós, adotou-se o modelo de energia proposto em [Silva et al. 2012]. Nesse modelo, o consumo de energia é classificado em cinco estados

- **Desligado** – não há consumo de energia, simula a situação em que a interface de rede está desligada.
- **Inativo** – consumo de energia reduzido, simula a situação em que a interface de rede está em modo de espera.
- **Scan** – o nó consome energia para descobrir outros nós vizinhos.
- **Transmitindo** – o nó consome energia enquanto envia uma mensagem.
- **Recebendo** – o nó consome energia enquanto está recebendo uma mensagem.

Nas simulações realizadas, assumiu-se que os nós incialmente têm um nível máximo de energia. Para efeito de simulação, considerou-se que o nível de energia dos nós é medido em unidades e inicialmente cada um deles tem 500 unidades de energia (nível máximo de energia). Assumiu-se também que os dispositivos são recarregados uma vez a cada 24 horas. O consumo de energia do dispositivo depende do estado e do número de operações usando a sua interface de rede. Por exemplo: se um nó está transmitindo, recebendo ou procurando vizinhos, assumiu-se que são reduzidas 25 unidades para cada envio/recebimento de mensagem e/ou para cada scaneamento realizado. Se o nó está no modo desligado ou inativo, assumiu-se que não existe consumo de energia.

5. Resultados obtidos

Nesta seção são apresentados os resultados obtidos ao executar o projeto de experimentos definido na seção anterior.

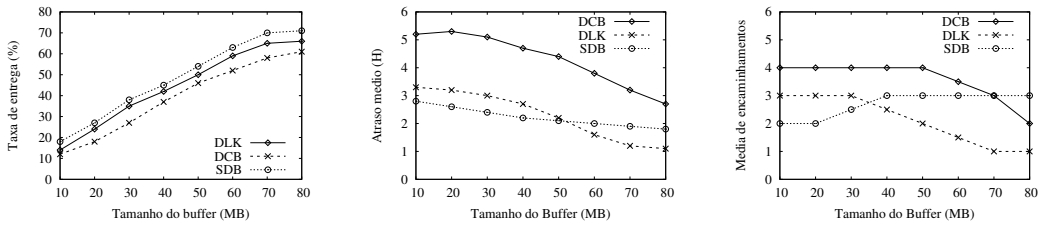

Figura 1. *Resultados para o trace de Cambridge*

5.1. Taxa de entrega

A taxa de entrega pode ser definida como a razão entre a quantidade de mensagens entregues pela quantidade de mensagens criadas na rede. As Figuras 1.(a) e 2.(a) apresentam os resultados obtidos para a métrica taxa de entrega nos cenários de mobilidade Cambridge e *Reality*. Com relação à métrica taxa de entrega, a hipótese levantada era a de que a consideração do egoísmo no gerenciamento do *buffer* dos nós poderia causar um impacto positivo no que diz respeito à quantidade de mensagens entregues na rede. Como pode-se observar nos resultados para à métrica taxa de entrega, o algoritmo SDB obteve o melhor resultado em ambos os cenários, entregando mais mensagens que as demais políticas testadas, validando a hipótese levantada.

A justificativa para esse desempenho está ligada diretamente à consideração da força da amizade como um subcritério para descartar mensagens do *buffer* dos nós nos casos de nós egoístas sociais e não egoístas. Em ambos os casos, a escolha de que mensagem deverá ser retirada do *buffer* é baseada na força da amizade entre o nó que está tomando a decisão e os destinatários das mensagens armazenadas no *buffer*. Como citado anteriormente, os nós que são amigos tendem a estarem em contato com uma frequência maior do que nós que não são amigos, favorecendo assim que mensagens sejam entregues entre si.

5.2. Atraso médio

O atraso médio pode ser definido como o tempo médio necessário para que uma mensagem alcance o seu destinatário. As Figuras 1.(b) e 2.(b) apresentam os resultados obtidos para a métrica atraso médio nos cenários de mobilidade Cambridge e *Reality*. Com relação à métrica atraso médio, a hipótese levantada era a de que a utilização do egoísmo no gerenciamento do *buffer* contribuiria para o aumento do tempo médio de entrega de uma mensagem. De acordo com os resultados apresentados para esta métrica, pode-se perceber que o algoritmo SDB alcançou o melhor desempenho no cenário de Cambridge para tamanhos de *buffer* menores que 60 MB, sendo superado pelo algoritmo DLK quando o tamanho de *buffer* foi maior ou igual a 60 MB. Porém, no cenário *Reality* o algoritmo SDB obteve o melhor resultado apresentando uma diferença de até duas horas a menos de atraso médio que os demais algoritmos testados. Portanto, a hipótese levantada mostrou-se falsa.

Fazendo-se uma análise do comportamento dos resultados obtidos pelo algoritmo SDB, percebe-se uma leve variação no seu desempenho em função do aumento do espaço em *buffer*. Por exemplo, no cenário de Cambridge a diferença entre o resultado para tamanho de *buffer* igual a 10 MB e o de 80 MB foi de apenas 1 hora. A partir dessa observação, é razoável supor que o atraso médio do algoritmo SDB sofre um leve impacto

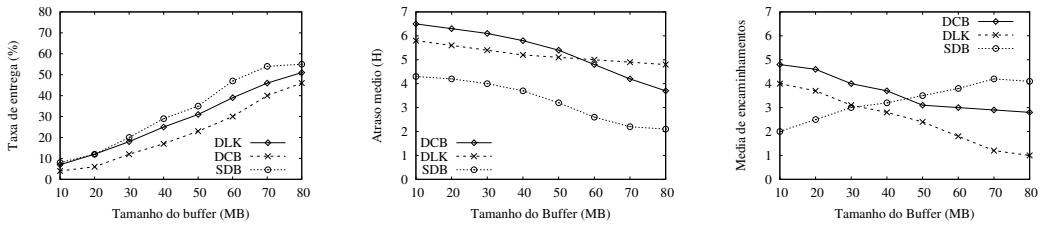

Figura 2. *Resultados para o trace Reality*

do aumento do espaço em *buffer*. Por outro lado, pode-se atribuir ao tempo entre contatos a razão para que o atraso médio do algoritmo SDB não sofra uma diminuição mais significativa mesmo com o aumento do espaço em *buffer*. O algoritmo SDB procura manter no *buffer* de um nó mensagens destinadas a outros nós com quem ele tem uma amizade forte, baseando-se na suposição de que esse nó é um dos melhores candidatos a entregar tais mensagens aos seus destinatários. No entanto, mesmo que aumente a probabilidade de entrega, essa decisão pode afetar o atraso médio de entrega caso o tempo entre contatos entre os nós seja alto. Nos resultados obtidos, pode-se concluir que o tempo entre contatos foi bastante parecido em qualquer tamanho de *buffer* testado, por isso o atraso médio para todos os tamanhos de *buffer* sofreu uma leve variação.

5.3. Média de saltos

A média de saltos pode ser definida como a quantidade média de encaminhamentos necessários para que uma mensagem alcance seu destinatário. As Figuras 1.(c) e 2.(c) apresentam os resultados obtidos para a métrica média de saltos nos cenários de mobilidade Cambridge e *Reality*. Com relação à métrica média de saltos, a hipótese levantada era a de que a utilização do egoísmo no gerenciamento do *buffer* contribuiria para a diminuição da média de saltos, porém, de acordo com os resultados obtidos, tal hipótese mostrou-se falsa.

Como pode-se observar, o algoritmo DLK obteve o melhor resultado com relação à métrica média de saltos para a maioria dos tamanhos de *buffer* testados em ambos os cenários. O algoritmo SDB obteve um resultado diferente dos demais algoritmos testados. Para os algoritmos DLK e DCB, quanto maior o espaço em *buffer*, menor a quantidade de saltos necessários para entregar a mensagem ao destinatário. O algoritmo SDB obteve um desempenho contrário a isto, ou seja, quanto maior o espaço em *buffer*, mais saltos foram necessários em média para que uma mensagem alcançasse seu destinatário. A justificativa é que com uma quantidade maior de espaço em *buffer*, mais mensagens podem ser armazenadas e encaminhadas em contatos posteriores e menos descartes são necessários. Dessa forma, mensagens que antes eram descartadas, agoram podem ser encaminhadas a outros nós que supostamente também sejam bons candidatos a entregar a mensagem ao destinatário. Apesar de essa prática aumentar a probabilidade de entrega, aumenta também a quantidade média de encaminhamentos necessários para que uma mensagem chegue ao seu destinatário.

6. Conclusões e Trabalhos Futuros

Neste trabalho, apresentou-se um algoritmo de gerência de *buffer* para redes DTN baseada no egoísmo dos nós. O objetivo principal deste algoritmo é tomar a decisão de que

mensagem descartar, em caso de *overflow*, baseando-se no tipo de egoísmo do nó, o qual pode ser classificado em individual, social ou não egoísta, e levando em consideração também a força da amizade entre os nós.

Para validar o algoritmo proposto, um conjunto de experimentos considerando métricas relacionadas à entrega de mensagens na rede foi realizado em dois cenários de mobilidade real bem conhecidos: Cambridge e *Reality*. Apesar de se levantarem algumas hipóteses negativas com relação ao uso do egoísmo no gerenciamento do *buffer* para DTNs, os resultados obtidos indicam que em ambos os cenários, o desempenho do algoritmo proposto mostrou-se muito promissor e inclusive superou outros algoritmos de gerenciamento existentes na literatura em termos da taxa de entrega e do atraso médio de entrega, além de obter um valor razoável de média de saltos.

Como trabalhos futuros, pretende-se avaliar de maneira mais extensa o desempenho do algoritmo SDB, comparando-o com outros algoritmos existentes na literatura para DTN. Pretende-se também avaliar o algoritmo SDB em conjunto com outros algoritmos de roteamento bem conhecidos para redes DTN, inclusive algoritmos de roteamento baseados em características sociais. Além disso, pretende-se inserir no algoritmo SDB uma forma de classificar o egoísmo dos nós de maneira dinâmica, coletando informações daquele nó e aplicando algoritmos de aprendizagem de máquina ou sistemas de reputação já existentes na literatura.

Referências

Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and Weiss, H. (2007). Delay-tolerant networking architecture. *RFC4838, April*.

de Oliveira, C. T., Moreira, M. D., Rubinstein, M. G., Costa, L. H. M., and Duarte, O. C. M. (2007). Redes tolerantes a atrasos e desconexões. *SBRC Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Domingos, P. and Pazzani, M. (1997). On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130.

Eagle, N. and Pentland, A. (2006). Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268.

Fall, K. (2003). A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34. ACM.

Gama, J. and teorema de Bayes, O. (2012). Aprendizagem bayesiana introdução. *Universidade do Porto*, pages 1–22.

Keränen, A., Ott, J., and Kärkkäinen, T. (2009). The ONE Simulator for DTN Protocol Evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA. ICST.

Krifa, A., Baraka, C., and Spyropoulos, T. (2008). Optimal buffer management policies for delay tolerant networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON'08. 5th Annual IEEE Communications Society Conference on*, pages 260–268. IEEE.

Lewis, D. D. (1998). Naive (bayes) at forty: The independence assumption in information retrieval. In *Machine learning: ECML-98*, pages 4–15. Springer.

Li, Q., Zhu, S., and Cao, G. (2010). Routing in socially selfish delay tolerant networks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE.

Lindgren, A. and Phanse, K. S. (2006). Evaluation of queueing policies and forwarding strategies for routing in intermittently connected networks. In *Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on*, pages 1–10. IEEE.

Naves, J. F., Moraes, I. M., and Albuquerque, C. V. (2012). Lps e lrf: Politicas de gerenciamento de buffer eficientes para redes tolerantes a atrasos e desconexões. *Simpósio Brasileiro de Redes de Computadores (SBRC?12)*, pages 293–305.

Pentland, A., Eagle, N., and Lazer, D. (2009). Inferring social network structure using mobile phone data. *Proceedings of the National Academy of Sciences (PNAS)*, 106(36):15274–15278.

Rashid, S., Ayub, Q., Zahid, M. S. M., and Abdullah, A. H. (2011). E-drop: An effective drop buffer management policy for dtn routing protocols. *International Journal*, 13(7):8–13.

Resta, G. and Santi, P. (2002). An analysis of the node spatial distribution of the random waypoint mobility model for ad hoc networks. In *Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 44–50. ACM.

Settawatcharawanit, T., Yamada, S., Haque, E., Rojviboonchai, K., et al. (2013). Message dropping policy in congested social delay tolerant networks. In *Computer Science and Software Engineering (JCSSE), 2013 10th International Joint Conference on*, pages 116–120. IEEE.

Silva, D. R., Costa, A., and Macedo, J. (2012). Energy impact analysis on dtn routing protocols. *ExtremeCom 2012*, pages 1–6.

Souza, C., Mota, E., Galvao, L., Manzoni, P., and Cano, J. C. (2014). Drop less known strategy for buffer management in dtn nodes. In *Proceedings of the Latin America Networking Conference on LANC 2014*, LANC '14, pages 6:1–6:7, New York, NY, USA. ACM.

Tang, L., Chai, Y., Li, Y., and Weng, B. (2012). Buffer management policies in opportunistic networks. *Journal of Computational Information Systems*, 8(12):5149–5159.

Vahdat, A., Becker, D., et al. (2000). Epidemic routing for partially connected ad hoc networks. Technical report, Technical Report CS-200006, Duke University.

Xia, F., Liu, L., Li, J., Ma, J., and Vasilakos, A. V. (2015). Socially aware networking: A survey. *IEEE Systems Journal*, 9(3):904–921.

Yoon, J., Liu, M., and Noble, B. (2003). Random waypoint considered harmful. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 1312–1321. IEEE.

Zhu, Y., Xu, B., Shi, X., and Wang, Y. (2013). A survey of social-based routing in delay tolerant networks: positive and negative social effects. *Communications Surveys & Tutorials, IEEE*, 15(1):387–401.