INSERINDO ASPECTOS SOCIAIS NA GERÊNCIA DE BUFFER PARA REDES TOLERANTES AO ATRASO E DESCONEXÕES

CAMILO BATISTA DE SOUZA

INSERINDO ASPECTOS SOCIAIS NA GERÊNCIA DE BUFFER PARA REDES TOLERANTES AO ATRASO E DESCONEXÕES

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas como requisito parcial para a obtenção do grau de Mestre em Informática.

Orientador: Dr. Ing. Edjair de Souza Mota

Manaus

Maio de 2013

de Souza, Camilo Batista
D1234p Inserindo aspectos sociais na gerência de buffer para redes tolerantes ao atraso e desconexões / Camilo Batista de Souza. — Manaus, 2013
xxiii, 70 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal do Amazonas
Orientador: Dr. Ing. Edjair de Souza Mota

1. Computação — Teses. 2. Redes — Teses. I. Orientador. II. Título.

CDU 519.6*82.10

[Folha de Aprovação]

Quando a secretaria do Curso fornecer esta folha,
ela deve ser digitalizada e armazenada no disco em formato gráfico.

Se você estiver usando o `pdflatex`,
armazene o arquivo preferencialmente em formato PNG
(o formato JPEG é pior neste caso).

Se você estiver usando o `latex` (não o `pdflatex`),
terá que converter o arquivo gráfico para o formato EPS.

Em seguida, acrescente a opção `approval={`*nome do arquivo*`}`
ao comando `\ppgccufmg`.

Se a imagem da folha de aprovação precisar ser ajustada, use:
`approval=[`*ajuste*`][`*escala*`]{`*nome do arquivo*`}`
onde *ajuste* ï£¡ uma distância para deslocar a imagem para baixo
e *escala* é um fator de escala para a imagem. Por exemplo:
`approval=[-2cm][0.9]{`*nome do arquivo*`}`
desloca a imagem 2cm para cima e a escala em 90%.

Dedico esse trabalho à minha família em especial à minha mãe, razão da minha vida.

Agradecimentos

Agradeço primeiramente à Deus pelo dom da vida e por ter me dado a oportunidade de fazer este trabalho. Toda honra e toda glória a Tí, Senhor. Agradeço especialmente à minha mãe. A razão de estar aqui hoje é você. Agradeço todos os dias a Deus por ter me dado esse presente que é ter você como mãe. Obrigado pela força, dedicação, carinho, incentivo e principalmente pelo exemplo que és pra mim. Tudo que tenho, tudo que sou, devo a você!

Agradeço à Géssica Vasconcelos por me suportar e entender todos os momentos que estive que ficar ausente por conta dos experimentos. Aos meus amigos Jonathan Pessoa, Nícolas Hatta e Damião Soares por todo o companheirismo e força repassadas nesse período.

Faço um agradecimento especial aos colegas de grupo JB, Poliani e principalmente ao Diogo por toda a ajuda dada nesse período. Aos colegas de curso Daniel Frazão, Helmer Mourão e Rafael Câmara e aos demais colegas também.

Gostaria de fazer um agradecimento especial ao meu orientador, professor Dr. Edjair Mota por ter me auxiliado e me orientado nesses anos, indicando os melhores caminhos a seguir. Obrigado, professor, pela paciência e pelos ensinamentos que certamente irei levar por toda a vida.

Por fim, a todos que direta ou indiretamente contribuíram para a realização desse trabalho o meu agradecimento e a minha sincera gratidão. Eis aqui a realização de um sonho e todos citados aqui de alguma forma ajudaram para que ele virasse realidade. Obrigado.

"O sofrimento é passageiro, desistir é para sempre."
(Lance Armstrong)

Resumo

Alguns tipos de Redes Tolerantes ao Atraso e Desconexões precisam da ajuda dos humanos para repassar dados entre os usuários. No entanto, humanos são socialmente egoístas e tendem a querer ajudar somente usuários com quem mantém uma relação social, fato que prejudica o desempenho da rede. Diante do espaço limitado de memória secundária nos dispositivos, a gerência de *buffer* se torna fator crucial para que o desempenho da rede seja satisfatório. Este trabalho apresenta uma nova política de gerência de *buffer* para DTNs, baseada na força da relação social entre os usuários.

Palavras-chave: Redes Tolerantes ao Atraso e Desconexões, relacionamento social, gerência de *buffer*. .

Abstract

Some types of Delay and Disruption Tolerant Networks need of humans help to send data between users. However, humans are socially selfishness and prefer to help users with whom they have a social relationship, fact that affects the network performance. Given the limited memory secondary space in the devices, the buffer management becomes a crucial factor to satisfactory network performance. This work presents a novel buffer management policy for DTNs, based on social relationship strength between users.

Keywords: Delay and Disruption Tolerant Networks, social relationship, buffer management.

Lista de Figuras

2.1 Exemplo 1: Internet Interplanetária. Nesta modalidade de Internet os atrasos na comunicação podem ser da ordem de horas e até mesmo dias 8
2.2 Exemplo 2: Comunidade Rural da região Amazônica. Nestes ambientes dificilmente encontra-se infraestrutura básica para o funcionamento da Internet 8
2.3 Funcionamento do protocolo TCP (Fonte: de Oliveira et al. [2007]) 9
2.4 Camada de agregação (fonte: Oliveira [2008]) 11

4.1 Funcionamento do algoritmo de roteamento Epidêmico quando dois nós A e B estão no mesmo raio de transmissão. (fonte: Vahdat et al. [2000]) . . . 23

5.1 Taxa de entrega utilizando roteamento Epidêmico no trace de *Cambridge* . 32
5.2 Taxa de entrega utilizando roteamento Epidêmico no trace *Reality* 33
5.3 Taxa de entrega de acordo com a força do laço social no trace *Reality* . . . 34
5.4 Taxa de entrega de acordo com a força do laço social no trace *Cambridge* . 35
5.5 Atraso Médio utilizando roteamento Epidêmico no trace de *Reality* 36
5.6 Atraso Médio utilizando roteamento Epidêmico no trace de *Cambridge* . . 37
5.7 Média de saltos utilizando roteamento Epidêmico no trace de *Reality* . . . 38
5.8 Média de saltos utilizando roteamento Epidêmico no trace de *Cambridge* . 39
5.9 Taxa de entrega utilizando roteamento *PROpHET* no trace de *Reality* . . 41
5.10 Taxa de entrega utilizando roteamento *PROpHET* no trace de *Cambridge* 42
5.11 Taxa de entrega de acordo com a força do laço social no trace *Reality* . . . 43
5.12 Taxa de entrega de acordo com a força do laço social no trace de *Cambridge* 44
5.13 Atraso Médio utilizando roteamento *PROpHET* no trace de *Cambridge* . . 45
5.14 Atraso Médio utilizando roteamento *PROpHET* no trace *Reality* 46
5.15 Média de Saltos utilizando roteamento *PROpHET* no trace de *Cambridge* 47
5.16 Média de Saltos utilizando roteamento *PROpHET* no trace *Reality* 48

Lista de Tabelas

1.1 Funcionamento de algumas políticas de gerenciamento de *buffer* que utilizam informações locais como base para o descarte 3

4.1 Conjuntos de dados utilizados no trabalho 22
4.2 Critérios utilizados para classificação da força dos laços sociais entre os nós no trace de *Cambridge* 26
4.3 Parâmetros utilizados nos experimentos 27

5.1 Média de saltos apresentado pela política DLK no trace *Cambridge* 40
5.2 Média de saltos apresentado pela política DLK no trace *Reality* 40
5.3 Média de saltos apresentado pela política DLK no trace *Reality* 49
5.4 Média de saltos apresentado pela política DLK no trace *Cambridge* 49

Sumário

Agradecimentos **ix**

Resumo **xiii**

Abstract **xv**

Lista de Figuras **xvii**

Lista de Tabelas **xix**

1 Introdução **1**

1.1 Motivação 2

1.2 Objetivos 2

1.3 Trabalhos relacionados 3

1.4 Organização do trabalho 6

2 Fundamentação Teórica **7**

2.1 Redes Tolerantes a Atrasos e Desconexões 7

2.1.1 Conceito 7

2.1.2 Arquitetura DTN 10

2.1.3 Tipos de Contato em uma DTN 11

2.1.4 Métricas usadas em DTNs 12

3 Descrição do Problema e Investigação Experimental **15**

3.1 Gerência de buffer utilizando relações sociais 17

3.1.1 Métrica força da relação social 17

3.1.2 Algoritmo de Gerência de buffer Drop Less Known 18

4 Metodologia **21**

4.1 Conjuntos de dados de mobilidade utilizados 21

4.2 Algoritmos de roteamento 22
4.2.1 Algoritmo Epidêmico 23
4.2.2 Algoritmo *PROpHET* 23
4.3 Classificação dos laços sociais 24
4.3.1 Classificação da força das relações sociais no trace Reality . . . 25
4.3.2 Classificação da força das relações sociais no trace de Cambridge 25
4.4 Métricas de avaliação 26
4.5 Políticas de gerenciamento de buffer utilizadas na avaliação de desempenho 27
4.6 Parâmetros gerais das simulações 27
4.7 Métodos Estatísticos 28
4.7.1 Método MSER-5 28
4.7.2 Método Overlapping Batch Means 29
4.7.3 Correlação de Von Neuman 29

5 Resultados **31**
5.1 Resultados dos Experimentos usando algoritmo Epidêmico 31
5.1.1 Taxa de Entrega 31
5.1.2 Atraso Médio de entrega 35
5.1.3 Média de Saltos 37
5.2 Resultados dos Experimentos usando algoritmo de roteamento *PROpHET* 40
5.2.1 Taxa de entrega 40
5.2.2 Atraso Médio 44
5.2.3 Média de Saltos 46

6 Conclusões **51**

Referências Bibliográficas **53**

Apêndice A Códigos auxiliares implementados na pesquisa **57**
A.1 Métodos estatísticos 57
A.1.1 Implementação do método MSER-5 57
A.1.2 Implementação do método OBM 60
A.1.3 Implementação da classe que elimina a correlação 62
A.2 Arquivos de configuração utilizados 63
A.2.1 Arquivo de configuração usado para o trace de *Cambridge* com o roteamento Epidêmico 63
A.2.2 Arquivo de configuração usado para o trace de *Cambridge* com o roteamento PROPhET 65

A.2.3 Arquivo de configuração usado para o trace de reality com o roteamento Epidêmico 67
A.2.4 Arquivo de configuração usado para o trace de reality com o roteamento PROPhET 69

Capítulo 1

Introdução

A pilha de protocolos TCP/IP, base da arquitetura da Internet, requer premissas básicas para o seu bom funcionamento, tais como existência de uma rota fim-a-fim, baixa perda de pacotes e baixa latência. No entanto, determinados ambientes de rede não fornecem algumas dessas premissas, fato que torna a operação desses protocolos inadequada e pouco robusta em tais ambientes de Oliveira et al. [2007]. Exemplos de tais ambientes são as comunidade rurais, redes de sensores sem fio, redes oportunistas, entre outros. Redes Tolerantes ao Atraso e Desconexões - DTNs foi o nome dado às redes de comunicação de dados que consideram estes aspectos.

Nessas redes, explora-se as eventuais conectividades criadas pela movimentação dos nós para carregar-se e transmitir-se mensagens Li et al. [2009], utilizando-se o mecanismo chamado de *store-carry-and-forward*. Através deste mecanismo, um nó torna-se capaz de armazenar uma mensagem no seu *buffer* de forma persistente para encaminhá-la ao destino ou a outros nós intermediários em contatos posteriores.

A utilização deste mecanismo tornou-se possível através da criação de uma sobrecamada denominada de *Bundle Layer* ou Camada de Agregação. Nessa camada é executado um protocolo denominado de protocolo de Agregação, o qual é responsável por indicar como serão realizadas as trocas de mensagens em DTNs. Para manter a interoperabilidade com qualquer outro tido de rede, essa camada foi colocada estrategicamente entre a camada de Aplicação e a camada de Transporte do modelo TCP/IP Oliveira [2008].

No entanto, apesar de possibilitar a comunicação, a combinação do mecanismo *store-carry-and-forward* com protocolos de replicação de mensagens pode levar a muitos cenários de transbordamento do *buffer* dos nós e causar o fenômeno chamado de *overflow*, o qual se refere à falta de espaço suficiente em *buffer* quando uma nova mensagem deve ser armazenada.

Diante deste cenário, o foco desta pesquisa foi o gerenciamento de *buffer* em DTNs considerando-se fatores do comportamento dos humanos como o relacionamento social e o egoísmo.

1.1 Motivação

Um dos maiores desafios em redes DTN é o gerenciamento do *buffer* dos nós. Em DTN, uma política de gerência de *buffer* define qual mensagem descartar em casos de *overflow* Naves [2012]. Na literatura DTN existem diversas propostas de políticas para gerenciamento do *buffer* dos nós, no entanto, segundo TANG et al. [2012], apesar de as políticas existentes na literatura melhorarem o desempenho de uma DTN em um determinado grau, todas têm suas limitações.

Recentemente, a consideração de características sociais forneceu um novo ângulo de visão no projeto de algoritmos para DTNs Zhu et al. [2012]. Segundo Zhu et al. [2012], grande parte dos algoritmos existentes para DTNs levam em consideração somente as oportunidades de contatos dentro da rede. No entanto, existem diversas características dos humanos que influenciam o desempenho dos algoritmos para DTNs, como por exemplo o egoísmo, as classes sociais, o altruísmo, entre outros. Desta maneira, novos algoritmos de roteamento têm sido propostos considerando algumas dessas características dos humanos, como em Okasha [2005], Chen & Kempe [2008], Xu et al. [2009].

A motivação para este trabalho é acreditar-se que uma dessas características dos humanos influencia na escolha de que mensagem descartar em casos de *overflow*. Essa característica é o egoísmo. De acordo com Li et al. [2010], no mundo real a maioria das pessoas são socialmente egoístas, ou seja, estão dispostas a transmitir mensagens para pessoas com quem mantém relações sociais, mas com outras não, e tal disposição varia de acordo com a força do laço social.

Considerando-se o egoísmo na gerência de *buffer* para DTNs, pode-se supor que ao ocorrer o transbordamento do *buffer*, usuários reais prefiram manter armazenadas nos seus dispositivos mensagens destinadas a usuários com quem se tem um laço social forte, ou seja, mensagens para outros usuários que por algum motivo desejam ajudar.

1.2 Objetivos

O objetivo deste trabalho é apresentar e validar uma proposta de gerenciamento do *buffer* para redes DTN considerando o comportamento egoísta dos usuários reais. Para

alcançar esse objetivo criou-se uma política de gerenciamento de *buffer* que utiliza como informação base para o descarte a força da relação social entre os usuários da rede. Para realizar-se a validação desse algoritmo, implementou-se um cenário de simulação em um simulador de eventos discretos para redes DTN proposto por Keränen et al. [2009], comparando-se os resultados com o de outras políticas de gerenciamento de *buffer* disponíveis na literatura.

1.3 Trabalhos relacionados

Na literatura sobre redes DTN, existem diversos trabalhos referentes ao gerenciamento do *buffer*. De acordo com TANG et al. [2012] as políticas de gerência de *buffer* existentes podem ser divididas em duas categorias: políticas baseadas em informações da rede e políticas baseadas em informações locais.

As políticas baseadas em informações locais, geralmente realizam apenas uma poda simples para realizar o descarte de uma mensagem. Estas políticas geralmente utilizam informações das mensagens que estão no *buffer* para tomar a decisão de descarte, como por exemplo o Tempo de vida da mensagem - TTL e o tamanho das mensagens. Exemplos de políticas pertencentes a esta classe são as seguintes: *drop random*, *drop least recently received*, *drop Oldest*, drop last, *drop Front*, *drop youngest* e *drop largest*. A Tabela 1.1 mostra uma breve descrição do funcionamento destas políticas.

Tabela 1.1. Funcionamento de algumas políticas de gerenciamento de *buffer* que utilizam informações locais como base para o descarte

Política de Gerência	**Mensagem descartada**
Drop Random (DR)	escolha aleatória
Drop Least Recently Received	mensagem mais antiga no *buffer*
Drop Oldest (DO)	mensagem com menor TTL
Drop Last (DL)	última mensagem na fila
Drop Front (DF)	primeira mensagem na fila
Drop Youngest (DY)	mensagem mais nova no *buffer*
Drop Largest (DLA)	maior mensagem no *buffer*

Em Zhang et al. [2007] uma análise de desempenho é feita com o algoritmo de roteamento Epidêmico Vahdat et al. [2000] e algumas políticas de gerência de *buffer*. Nesse trabalho, as políticas utilizadas foram *drop Tail*, *drop Head* e *drop Head* com Pri-

oridades. Os resultados mostram que *drop Head* com Prioridades tem um desempenho similar à situação de *buffer* infinito no cenário avaliado.

A política *drop largest* - DLA é proposta em Rashid & Ayub [2010]. Nessa política, quando ocorre *overflow* no *buffer*, a mensagem de maior tamanho é escolhida para o descarte. O trabalho apresenta uma comparação de DLA com a política *drop oldest* - DO e os resultados mostram que DLA supera DO. Soares et al. [2009] realizam uma avaliação de um esquema de gerência de *buffer* baseado no TTL da mensagem com o algoritmo de roteamento Epidêmico e *Spray and Wait* Spyropoulos et al. [2005] em uma rede veicular. Os resultados obtidos mostram que a utilização destes algoritmos de roteamento com a estratégia de descarte proposta, supera os algoritmos de roteamento *MaxProp* Burgess et al. [2006] e *PRoPHET* Lindgren et al. [2003], que têm suas próprias estratégias de descarte.

Li et al. [2009] propõem um mecanismo de gerência de *buffer* baseado na quantidade de replicações de uma mensagem. Ao ocorrer *overflow*, a política denominada de N-*Drop* realiza uma verificação em cada mensagem no *buffer* para encontrar uma ou mais mensagens que foram encaminhadas N vezes. Caso não exista nenhuma mensagem encaminhada N vezes, a última mensagem no *buffer* é descartada. Utilizando modelo *Random Waypoint* Hyytiä et al. [2005] o mecanismo de gerência de *buffer* é avaliado. Os resultados obtidos apresentam uma diminuição na média de saltos e um aumento na taxa de entrega.

Ayub & Rashid [2010] propõem uma política de gerência de *buffer* que utiliza um parâmetro chamado de *Threashhold*(T) para escolher mensagens a serem descartadas do *buffer*. Este valor de T representa um limiar referente ao tamanho da mensagem e, em caso de *overflow*, o algoritmo primeiramente verifica se a mensagem candidata a entrar no *buffer* está dentro deste limiar. Caso não esteja, nenhuma mensagem é retirada do *buffer*. Os resultados obtidos mostram que T-*Drop* reduz o descarte de mensagens, média de saltos e a sobrecarga na rede, enquanto aumenta a probabilidade de entrega, em comparação com *Drop Largest*.

Algumas políticas de gerência de *buffer* existentes, utilizam informações da rede para tomar a decisão de descarte. No entanto, com a frequente mudança de topologia da rede, é difícil obter informações completas do estado da mesma TANG et al. [2012]. Desta forma estas políticas utilizam informações parciais da rede. Um exemplo é Lindgren & Phanse [2006] que realizam uma avaliação da combinação de diversas estratégias de roteamento e políticas de gerência de *buffer*. As políticas de gerência de *buffer* utilizadas foram: *First in first out* - FIFO, *Evict most forwarded first* - MOFO, *Evict most favorably forwarded first* - MOPR, *Evict shortest life time first* - SHLI, e *Evict least probable first* - LEPR. Com exceção de FIFO, as outras políticas utilizadas

são baseadas em alguma informação retirada da rede, como exemplo o número de vezes que uma mensagem foi encaminhada pela rede e probabilidade de entrega. Os resultados obtidos demonstraram que a escolha da mensagem a ser roteada ou descartada é tem melhor desempenho ao utilizar-se previsibilidade, ao invés de apenas uma poda aleatória.

Em Krifa et al. [2008], uma política de gerência de *buffer* denominada *Global Knowledge based Drop*(GBD), é proposta. Essa política é baseada na disseminação de mensagens por encontros. A ideia é utilizar as informações dos encontros e estatística de aprendizado para aproximar o conhecimento global necessário para a política. Entretanto, por utilizar informações globais do estado da rede, a implementação da política torna-se complexa.

Em TANG et al. [2012], Lijun et. al utilizam a frequência média de encontros entre dois pares de nós para escalonar mensagens a serem enviadas e o número de réplicas de uma mensagem para tomar a decisão de descarte. Os resultados dos experimentos mostram que a política pode beneficiar a melhoria do desempenho de redes DTN em termos da taxa de entrega, atraso médio e da média de saltos.

Liu et al. [2011] utilizam a lei de diminuição marginal da utilidade para propor um esquema de gerência de *buffer* baseado no estado estimado de mensagens, como por exemplo o número de cópias na rede e a velocidade de disseminação de uma mensagem. Os resultados obtidos mostram uma melhor proporção de entrega e uma sobrecarga inferior a outros métodos testados.

Utilizando a ideia de que uma rede DTN pode ser utilizada para suportar diversas aplicações simultaneamente, e cada aplicação pode ter uma prioridade diferente, Fathima & Wahidabanu [2008] propõem um esquema de gerência de *buffer* com prioridades, em que o objetivo é eliminar do *buffer*, em casos de *overflow*, dados com prioridades inferiores. Os resultados obtidos com esta política apresentam melhoria no desempenho para roteamento Epidêmico com entrega preferencial.

Fischer et. al propõem duas políticas de gerência de *buffer* em Naves [2012]. Essas políticas foram denominadas por *Less Probable Sprayed* (LPS) e *Least Recently a Forwarded* (LRF). A política LPS utiliza a quantidade de replicações de uma mensagem juntamente com a probabilidade de entrega para decidir que mensagem descartar. A política LRF, por sua vez, descarta a mensagem encaminhada há mais tempo. Os resultados obtidos por ambas as políticas LPS e LRF apresentam um aumento na taxa de entrega, além de uma menor sobrecarga da rede.

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 os algoritmos de gerência de *buffer* para as mesmas. De acordo com TANG et al.

[2012], as políticas existentes na literatura podem melhorar o desempenho da rede em um determinado grau, mas todas têm suas limitações. Neste intuito, este trabalho apresenta uma nova proposta de gerência de *buffer* que utiliza o relacionamento social entre os usuários da rede como informação base para o descarte.

1.4 Organização do trabalho

O restante deste trabalho está organizado da seguinte forma: O Capítulo 2 apresenta a fundamentação teórica, usada no desenvolvimento do trabalho com os conceitos básicos de DTNs. No Capítulo 3 é feita uma contextualização do problema e de que maneira foi feita a investigação experimental. No Capítulo 4 é apresentada a metodologia para avaliação do algoritmo proposto e os resultados obtidos são apresentados no Capítulo 5. Finalmente no Capítulo 6 são apresentadas as conclusões e as propostas para a continuidade do presente trabalho.

Capítulo 2

Fundamentação Teórica

Para orientação e compreensão da aplicação da teoria envolvida na pesquisa desenvolvida no presente trabalho, é necessário o estudo das Redes Tolerantes a Atrasos e Desconexões. Nas seções subsequentes serão apresentados conceitos básicos e uma abordagem sucinta de cada tópico.

2.1 Redes Tolerantes a Atrasos e Desconexões

2.1.1 Conceito

A Internet, uma das maiores inovações tecnológicas das últimas décadas, tornou-se um dos principais meios de comunicação no mundo. Através desta tecnologia, tornou-se possível o compartilhamento de informações e o desenvolvimento de diversas aplicações que facilitam serviços, auxiliam e oferecem entretenimento à população mundial. Grande parte desse sucesso deve-se à operação dos protocolos da pilha TCP/IP, os quais são base da arquitetura da Internet.

Segundo Oliveira [2008], esse conjunto de protocolos foi projetado para funcionar sob condições típicas de redes cabeadas, e, assim, necessitam de algumas premissas básicas para o seu bom funcionamento, tais como a existência de uma rota fim-a-fim entre nó fonte e destino, baixa perda de pacotes, baixa latência, entre outros.

No entanto, determinados ambientes de rede não fornecem algumas dessas premissas, fato que torna a operação desses protocolos inadequada e pouco robusta em tais ambientes de Oliveira et al. [2007]. Convencionou-se chamar os ambientes que consideram estes aspectos por Redes Tolerantes ao Atraso e Desconexões (*Delay and Disruption Tolerant Networks* - DTNs) Fall [2003]. Exemplos de tais ambientes são redes de comunicações sem fio, comunicações entre dispositivos móveis, comunicações entre

dispositivos com restrições de energia, comunicações rurais, comunicações submarinas e comunicações interplanetárias Farrell et al. [2006]. As figuras 2.1 e 2.2 exemplificam alguns desses ambientes.

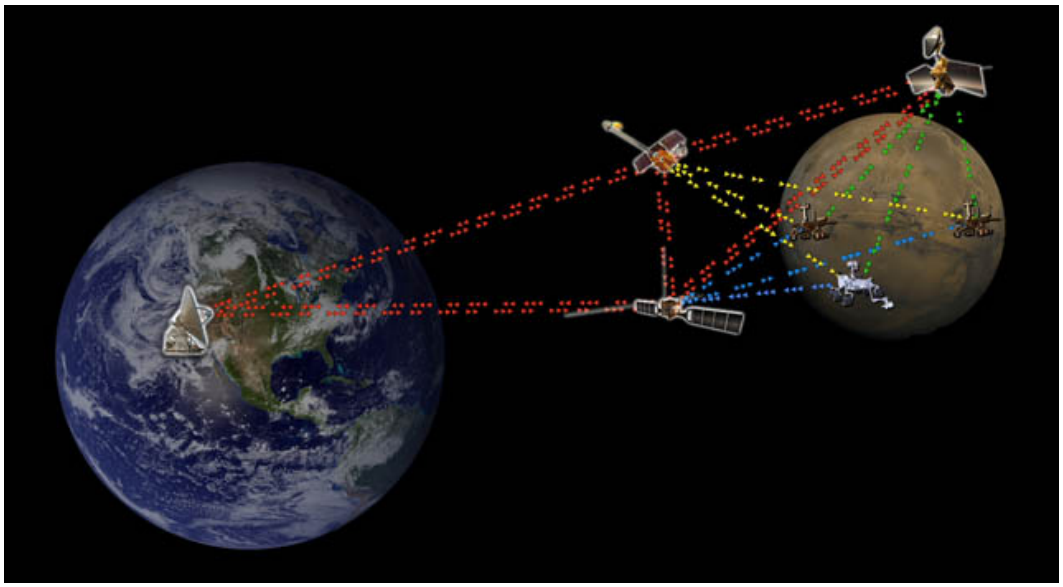

Figura 2.1. Exemplo 1: Internet Interplanetária. Nesta modalidade de Internet os atrasos na comunicação podem ser da ordem de horas e até mesmo dias

Figura 2.2. Exemplo 2: Comunidade Rural da região Amazônica. Nestes ambientes dificilmente encontra-se infraestrutura básica para o funcionamento da Internet

Os aspectos comuns a todas as DTNs dificultam principalmente a operação do protocolo *Transmission Control Protocol* - TCP. O TCP é um protocolo orientado a co-

nexão que garante confiabilidade na entrega de dados fim-a-fim de Oliveira et al. [2007]. O funcionamento deste protocolo é baseado em três etapas, que são: estabelecimento de conexão, transferência de dados e desconexão. Estas etapas são exemplificadas na figura 2.3.

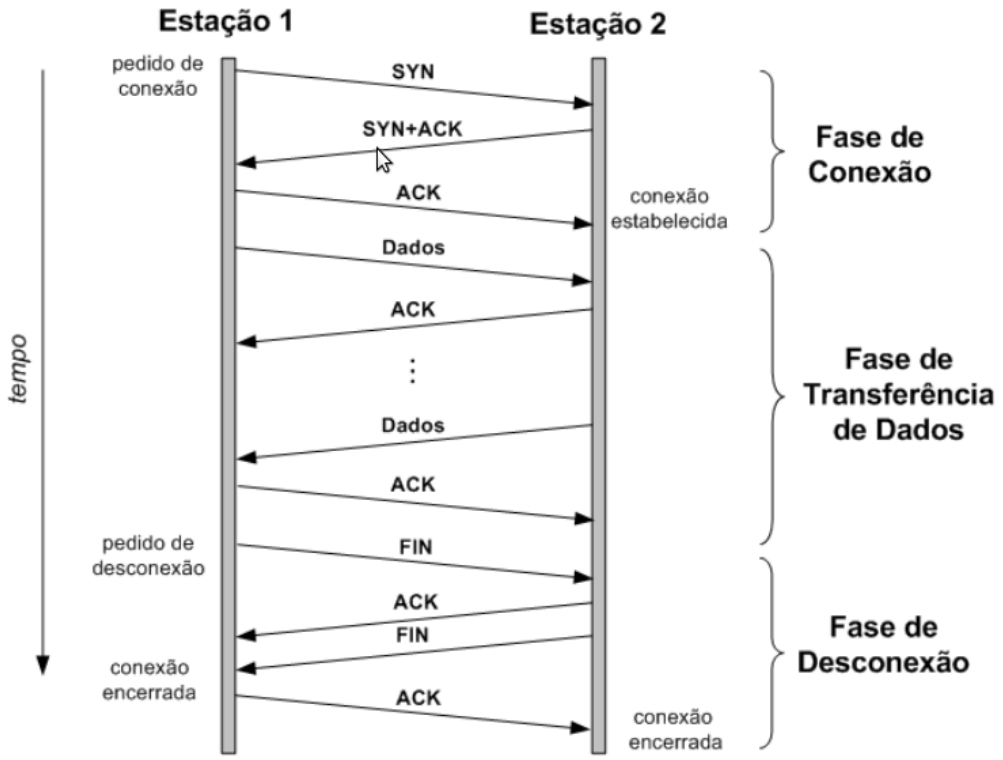

Figura 2.3. Funcionamento do protocolo TCP (Fonte: de Oliveira et al. [2007])

Segundo Postel [2003], durante todas as etapas da operação do TCP, mensagens são trocadas para controle e servem como indício de que os dados estão sendo entregues corretamente. A etapa de estabelecimento de conexão é feita através do mecanismo denominado de *three-way handshake*, que se trata da troca de três mensagens entre o nó fonte e o nó destino, indicando que a conexão foi estabelecida de forma correta. Na etapa de transferência de dados, sinais são trocados pelos nós envolvidos na comunicação indicando o recebimento completo de pacotes de dados. Estes sinais são denominados por *acknowledgments* - ACKs. Por fim, a etapa de desconexão realiza-se com a troca de quatro mensagens.

É fácil notar que para o funcionamento correto do procotolo TCP, faz-se necessário um *link* de conexão fim-a-fim entre os nós envolvidos na comunicação, por conta da troca de mensagens em cada uma das suas etapas. Dessa forma, o protocolo TCP

falha em ambientes onde tal premissa não é disponibilizada e novas estratégias devem ser utilizadas para possibilitar a comunicação.

No caso das redes DTN, criou-se uma arquitetura alternativa, cujo objetivo é superar os desafios gerados pela ausência das premissas básicas para o funcionamento da pilha TCP/IP. Tal arquitetura será apresentada na próxima seção.

2.1.2 Arquitetura DTN

A arquitetura DTN tem como principal característica suprir as dificuldades criadas pela ausência das premissas básicas para o funcionamento dos protocolos da Internet, possibilitando que, nos ambientes com tais dificuldades, a comunicação ocorra.

Tal arquitetura surgiu como uma derivação do projeto da Agência Espacial Americana - NASA, denominado Internet Interplanetária - IPN. Esse projeto tinha como objetivo criar uma arquitetura de redes que tornasse possível a existência de interoperabilidade entre a Internet terrestre e uma Internet em outros planetas e astronaves IPN [2013]. A principal questão a ser vencida nesta arquitetura era que a comunicação entre essas redes poderia sofrer grandes atrasos que poderiam durar horas e até mesmo dias. De acordo com as características deste problema, percebeu-se que a arquitetura proposta para o projeto IPN também poderia ser aplicada em ambientes terrestres onde os atrasos na comunicação são comuns.

A arquitetura DTN é então definida no documento RFC 4838, que descreve como um conjunto de nós se organiza para armazenar e encaminhar mensagens em ambientes sujeitos a atrasos longos e/ou variáveis e com freqüentes desconexões Cerf et al. [2007].

Em casos de atrasos na comunicação os nós de uma DTN utilizam um mecanismo denominado de armazenamento persistente para realizar o armazenamento de uma mensagem. Neste caso, em situações de desconexão ou atraso na comunicação, o armazenamento persistente possibilita o armazenamento integral da mensagem que deseja-se enviar, fato que permite o encaminhamento de tal mensagem em contatos posteriores.

Este mecanismo torna-se possível em DTNs com a criação de uma sobrecamada na pilha TCP/IP, denominada de Camada de Agregação (*Bundle Layer*). Esta sobrecamada foi colocada estrategicamente entre a camada de aplicação e a camada de transporte, com o objetivo de manter a interoperabilidade de uma DTN com qualquer tipo de rede que utilize a pilha de protocolos TCP/IP. Uma breve ilustração pode ser visualizada na figura 2.4.

Por utilizar da técnica do armazenamento persistente na ausência de um caminho completo até o destino para encaminhar a mensagem em contatos posteriores, diz-se que

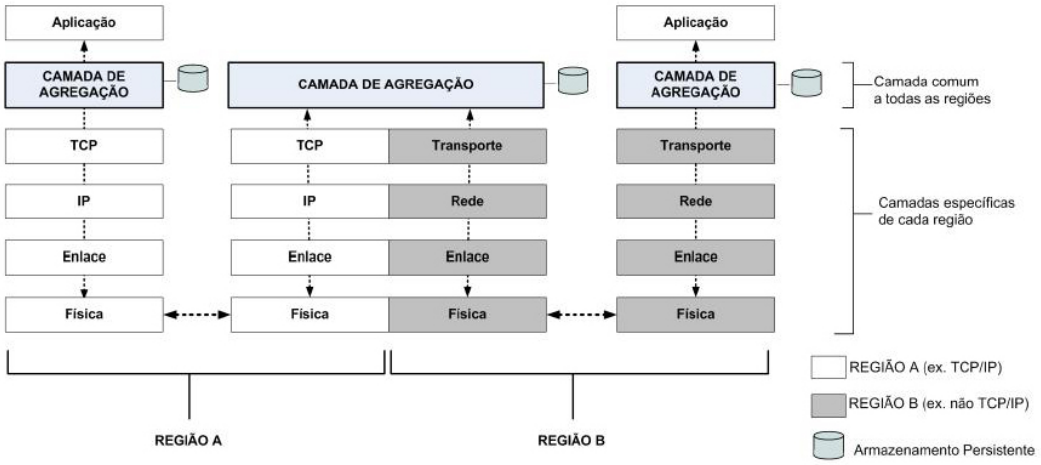

Figura 2.4. Camada de agregação (fonte: Oliveira [2008])

as DTNs são redes do tipo armazena-carrega-e-encaminha (*store-carry-and-forward*), ou seja, primeiramente recebem totalmente a mensagem, armazenam persistentemente e somente depois enviam para o destino ou para nós intermediários Oliveira [2008].

Além da função principal que é possibilitar o armazenamento persistente, nessa camada também opera o protocolo de Agregação. O protocolo de Agregação, por sua vez, é definido na RFC 5050 Scott & Burleigh [2007], onde são descritos os formatos de blocos e um resumo da descrição do serviço de troca de pacotes em redes DTN. Em DTNs uma mensagem recebe o nome de *Bundle* e pode ter tamanhos variados, recebendo assim o nome de *Application Data Units* - ADUs. Os *bundles* são criados pela camada de Agregação e transformados em uma ou mais Unidades de dados de protocolo - PDUs, ficando assim disponíveis para o armazenamento e encaminhamento. Essas unidades de dados são transmitidas quando ocorre o fenômeno do contato entre dois nós.

Dada a importância do evento do contato, a seção a seguir aborda o conceito e os tipos de contatos em uma rede DTN.

2.1.3 Tipos de Contato em uma DTN

Segundo Warthman [2007], um contato em DTN corresponde a uma ocasião favorável a troca de dados entre dois nós. Tal evento ocorre quando ambos os nós estão no raio de alcance do outro, possibilitando que os dados armazenados, dependendo da estratégia de roteamento sejam transferidos.

Na arquitetura DTN são definidos 5 tipos de contatos: persistentes, em demanda, agendados, oportunistas e previsíveis. A seguir uma breve descrição de cada um destes

tipos de contatos.

- persistente - segundo Neto [2012], são contatos que estão sempre disponíveis. Por exemplo, uma conexão cabeada.
- em demanda - são contatos que, para ocorrerem, necessitam que uma ação seja tomada. Segundo Oliveira [2008], uma conexão discada, do ponto de vista do usuário, serve como exemplo de contato sob demanda.
- agendado - em DTNs os nós podem estabelecer o momento em que farão contato para troca de informações. Dessa forma, este contato é denominado de agendado ou ainda segundo Oliveira [2008], programado. Um exemplo deste tipo de contato em redes DTNs reais são as Redes Interplanetárias.
- oportunista - são contatos que ocorrem a medida que os nós se movimentam e entram no mesmo raio de ação, surgindo assim a oportunidade de troca de informações.
- previsível - são contatos baseados em históricos de contatos anteriores. Levando em conta essa informação, os nós podem fazer uma previsão de que momento ocorrerá o contato com determinado nó da rede.

Para avaliar-se o desempenho de uma DTN, são necessárias métricas que mostrem a eficiência e a eficácia da rede. Dessa forma, a seguir são apresentadas algumas das métricas mais utilizadas em DTNs.

2.1.4 Métricas usadas em DTNs

Para medir o desempenho de uma DTN, algumas métricas são utilizadas. Segundo Keränen et al. [2009], as métricas mais usadas neste sentido são:

- Tempo de contato - Mede o tempo médio de contato entre os nós da rede
- Tempo entre contatos - Mede o tempo médio entre os contatos dos nós da rede.
- Taxa de entrega - Razão entre o número de mensagens recebidas e o número de mensagens enviadas.
- Atraso de Entrega - Intervalo de tempo médio entre o envio e o recebimento das mensagens na rede.

- Ocupação da rede - Porcentagem média de ocupação de espaço de armazenamento dos nós da rede.

Com relação ao gerenciamento do *buffer*, tema abordado neste trabalho, destacam-se outras duas métricas:

- Atraso médio de entrega - é definido como o intervalo de tempo médio entre o envio e o recebimento das mensagens na rede.
- Média de saltos - é a quantidade média de saltos de uma mensagem até ser entregue ao destino.

A seguir, será definido o problema abordado neste trabalho e a investigação experimental realizada.

Capítulo 3

Descrição do Problema e Investigação Experimental

Apesar de existir um consenso da comunidade científica sobre a arquitetura utilizada nas DTNs Cerf et al. [2007], ainda existem diversos problemas nessa modalidade de comunicação que continuam em aberto e são alvos de diversas pesquisas no mundo todo. Dentre os principais tópicos ainda em aberto na literatura DTN, pode-se destacar os problemas a seguir como os mais pesquisados atualmente:

- roteamento de Mensagens: em redes DTN uma rota fim-a-fim entre dois nós pode sofrer com quebras de conexão ou até mesmo nunca existir. Dessa maneira, quando deseja-se encaminhar uma mensagem a um determinado nó na rede, rotas entre a origem e o destino devem ser traçadas, aproveitando-se as oportunidades de contato, para garantir o sucesso na comunicação. O roteamento é um desafio comum a todas as DTNs, pois, os protocolos que operam nesse tipo de rede devem estar aptos a superar os longos atrasos e frequentes desconexões Oliveira & Duarte [2007]. O problema do roteamento de mensagens em DTNs consiste então em traçar rotas de um nó fonte até um destino considerando os atrasos e desconexões comuns nessas redes.

- gerência de energia: a capacidade de energia dos nós é um recurso bastante crítico em redes DTNs. Segundo Azevedo [2010], o consumo de energia ocorre principalmente no processo de descoberta de dispositivos vizinhos e a economia eficiente deste recurso é um dos principais desafios em redes DTN. Os dispositivos que compõe a rede geralmente são equipados com baterias de energia que tem uma duração limitada. Assim, à medida que os recursos do dispositivo são utilizados, a capacidade de energia diminui. O problema da gerência de energia consiste na

utilização de tal recurso da forma mais eficiente possível, de forma a aumentar a autonomia da bateria dos nós e o tempo de vida operacional da rede.

- gerência de *buffer*: assim como a energia, o espaço em *buffer* nos nós de redes DTN é um recurso escasso e limitado. Segundo Li et al. [2009], para incrementar-se a probabilidade de entrega, o roteamento em DTNs pode requerer que várias cópias de uma mesma mensagem sejam replicadas na rede. No entanto, combinando-se estratégias de roteamento baseadas em replicações com o espaço limitado no *buffer* dos nós, pode-se chegar a cenários onde uma nova mensagem deve ser armazenada, mas o espaço em *buffer* é insuficiente. Em DTNs esse fenômeno é chamado de *overflow*. Quando ocorre o *overflow*, uma ou mais mensagens devem ser selecionadas para retirar-se do *buffer*, permitindo armazenar-se uma nova mensagem. O problema da gerência de *buffer* consiste na seleção das mensagens que serão retiradas do *buffer* em casos de *overflow*. Essa escolha é de suma importância para maximizar a performance do sistema pois a mesma influencia diretamente o roteamento de mensagens e, consequentemente, a entrega de mensagens na rede.

É evidente que o bom funcionamento de uma DTN requer um equilíbrio na utilização dos recursos da rede. Diante deste desafio, devem ser projetados algoritmos que gerenciem da melhor maneira tais recursos. Atualmente, diversas propostas de algoritmos de roteamento são encontradas na literatura, como em Lindgren et al. [2003],Vahdat et al. [2000], Spyropoulos et al. [2005],Oliveira & de Albuquerque [2009]. Da mesma forma, existem diversas propostas de políticas para o gerenciamento do *buffer* como em Fathima & Wahidabanu [2008], Krifa et al. [2008], Ayub & Rashid [2010], Naves [2012] e ainda diversos trabalhos relacionados ao gerenciamento de energia, como em Juang et al. [2002], Jun [2007], Jun et al. [2006]. No entanto, ainda não existe consenso com relação a estes tópicos de pesquisa.

Com relação ao problema do gerenciamento do *buffer*, TANG et al. [2012] afirmam que, apesar de as políticas existentes na literatura melhorarem o desempenho de uma DTN em um determinado grau, todas têm suas limitações.

Algumas estratégias de gerência de *buffer* realizam uma poda simples nas mensagens locais do *buffer* para selecionar a mensagem que será descartada, enquanto outras políticas utilizam uma ou mais informações da rede para realizar tal seleção.

Recentemente, a consideração de características sociais forneceu um novo ângulo de visão no projeto de algoritmos para DTNs Zhu et al. [2012]. Segundo Zhu et al. [2012], grande parte dos algoritmos existentes para DTNs levam em consideração so-

mente as oportunidades de contatos dentro da rede. No entanto, existem diversas características dos humanos que influenciam o desempenho dos algoritmos para DTNs, como por exemplo o egoísmo, as classes sociais, o altruísmo, etc. Neste sentido, novos algoritmos têm sido propostos considerando algumas dessas características dos humanos, como em Okasha [2005], Chen & Kempe [2008], Xu et al. [2009].

A investigação experimental deste trabalho partiu do problema do gerenciamento do *buffer* para DTNs, baseando-se na ideia de que uma das características dos humanos citadas anteriormente influencia na escolha de que mensagem descartar em casos de *overflow*. Essa característica é o egoísmo.

De acordo com Li et al. [2010], no mundo real a maioria das pessoas são socialmente egoístas, ou seja, estão dispostas a transmitir mensagens para pessoas com quem mantém relações sociais, mas com outras não, e tal disposição varia de acordo com a força do laço social.

Considerando-se o egoísmo na gerência de *buffer* para DTNs, pode-se supor que ao ocorrer o *overflow*, usuários reais prefiram manter armazenadas nos seus dispositivos mensagens destinadas a usuários com quem se tem um laço social forte, ou seja, mensagens para aqueles usuários que por algum motivo desejam ajudar.

3.1 Gerência de buffer utilizando relações sociais

Baseando-se nos fatos citados anteriormente, propõe-se neste trabalho uma política para gerência de *buffer* de DTNs baseada no relacionamento social entre os usuários da rede. O principal objetivo desta política é priorizar o armazenamento de mensagens destinadas a usuários com quem se tem maior relação social e descartar mensagens destinadas a usuários menos conhecidos. Por este motivo, denominou-se essa política por *Drop Less Known* - DLK. Neste sentido, adotou-se uma nova métrica chamada de *força da relação social*, que é utilizada como base para o descarte feito pela política DLK. A seguir, maiores detalhes sobre esta métrica serão apresentados.

3.1.1 Métrica força da relação social

No mundo real, dois indivíduos podem ser da mesma família, podem ter um laço de amizade, podem ser apenas conhecidos ou também podem ser desconhecidos. Considerando o comportamento egoísta dos humanos, pode-se afirmar que, de maneira geral, os usuários reais tendem a ajudar primeiramente usuários da sua família ou amigos. De acordo com Li et al. [2010], os usuários reais ajudam primeiramente pessoas com quem tem maior relação social por terem sido ajudados por elas anteriormente ou por

acreditarem que futuramente podem ser ajudados por estas pessoas. Além disso, os usuários reais tendem a não querer carregar nos seus dispositivos mensagens destinadas a outros usuários que sejam desconhecidos ou usuários que, por algum motivo, não desejam ajudar.

Dessa forma, neste trabalho adotou-se a métrica *força da relação social* para indicar-se a força do laço social entre dois pares de nós dentro da rede, fato que possibilita determinar a prioridade no descarte de mensagens em casos de *overflow*, servindo assim como base para o algoritmo DLK.

Neste sentido, para melhor representar-se a força dos laços sociais entre dois nós, utilizou-se três tipos de laços: laço forte, laço médio, laço fraco. O algoritmo DLK prioriza o descarte das mensagens pertencentes ao nós com quem se mantém os laços mais fracos.

No capítulo 4 são apresentado os procedimentos utilizados para definir a força dos laços sociais entre dois pares de nós neste trabalho. A seguir, maiores detalhes serão apresentados sobre o algoritmo de gerenciamento do *buffer* proposto.

3.1.2 Algoritmo de Gerência de buffer Drop Less Known

Como citado na seção anterior, o algoritmo DLK prioriza sempre o descarte de mensagens destinadas a usuários menos conhecidos, ou seja, para aqueles com os quais os quais a força do laço social é mais fraco. A seguir, apresenta-se o pseudo-código da política de gerenciamento de *buffer* DLK.

Algorithm 1 Algoritmo Drop Less Known

```
1:  procedure DLK(m, buffer[])
2:      strengthMin ← 0
3:      if m.rel.Strength >strengthMin then
4:          strengthMin ←m.rel.Strength
5:          for i = 1 to buffer.size() do
6:              if buffer[i].rel.Strength <strengthMin then
7:                  strengthMin ←buffer[i].rel
8:                  id ←buffer[i].id
9:              end if
10:         end for
11:     else
12:         id ←m.id
13:     end if
14: end procedure
```

Em caso de *overflow*, DLK inicia um procedimento de verificação para identificar

que mensagem deve ser retirada do *buffer*. No entanto, primeiramente verifica-se a força da relação social do destino da mensagem candidata a entrar no *buffer*. Se essa relação for fraca a política descarta a própria mensagem candidata. Senão, a política seleciona a mensagem presente no *buffer* destinada ao usuário menos conhecido, ou seja, aquele com quem a força do laço social é mais fraco. Se existir empate nessa escolha, a política seleciona para descartar a primeira dessas mensagens encontrada no *buffer*.

Para validar-se a política DLK, efetuaram-se simulações e comparou-se os resultados obtidos com o de outras políticas de gerenciamento do *buffer* disponíveis na literatura para DTNs. Os capítulos a seguir apresentam a metodologia utilizada nesta validação e os resultados obtidos.

Capítulo 4

Metodologia

Para validar a política de gerência de *buffer* baseada em relacionamento social proposta para redes tolerantes a atrasos e desconexões, optou-se pela técnica da simulação. Dessa forma, implementou-se um modelo de simulação que inclue a política DLK e outras políticas de gerenciamento de *buffer* disponíveis na literatura, permitindo quantificar-se o desempenho da política proposta. Esse cenário foi implementado no simulador *Opportunistic Network Environment* - ONE Keränen et al. [2009], onde experimentos visando avaliar métricas de entrega de mensagens na rede foram avaliadas.

Nas simulações realizadas neste trabalho, optou-se pela utilização de dois conjuntos de dados referentes à mobilidade humana disponíveis na literatura para DTNs. Os traces utilizados foram *Cambridge* e *Reality*. Como algoritmos de roteamento, optou-se pela utilização dos protocolos Epidêmico e *PROpHET* que são baseados em replicações de mensagens. Em cada rodada de simulação efetuada, utilizou-se técnicas para eliminação do transiente da simulação, técnica para o tratamento da correlação dos dados e para garantir que as coletas realizadas estivessem dentro de um intervalo de confiança de 95% de certeza.

As seções a seguir apresentam maiores detalhes sobre os conjuntos de dados, algoritmos de roteamento e todas as técnicas para aumentar a credibilidade das simulações utilizado neste trabalho.

4.1 Conjuntos de dados de mobilidade utilizados

Para representar a mobilidade humana, utilizou-se dois conjuntos de dados denominados por *Cambridge* Diot et al. [2004] e *Reality* Eagle & Pentland [2006]. Estes conjuntos de dados foram selecionados por serem bastante citados na literatura para DTNs, servindo como base para diversas pesquisas no mundo todo, além de representarem de

forma mais próxima a mobilidade dos humanos em ambientes reais.

O trace de *Cambridge* recebeu este nome por ter sido gerado na Universidade de *Cambridge* com a contribução de dois grupos de estudantes do laboratório de computação, principalmente alunos do curso de graduação e de alguns alunos dos cursos de mestrado e doutorado. Os estudantes carregavam dispositivos denominados *iMotes*. Ao todo foram utilizados 19 estudades do laboratório de computação que utilizaram 54 dispositivos para realizar as coletas que geraram o trace. O trace tem a duração de 11 dias.

Por sua vez, o trace *Reality* foi gerado a partir de um experimento realizado no período de 2004 a 2005 no *Massachusetts Institute of Technology* - MIT, utilizando um grupo de 100 usuários. Foram disponibilizadas para a comunidade científica informações de comunicação, proximidade, localização e informações de atividades destes usuários. Além da representação da mobilidade real, outro fator que serviu como motivação para utilização deste trace é o histórico de ligações entre os usuários que participaram dos experimentos. Esta informação foi utilizada como base para classificar-se a força da relação social entre dois nós na rede.

A tabela 4.1 apresenta alguns detalhes de ambos os traces utilizados neste trabalho.

Tabela 4.1. Conjuntos de dados utilizados no trabalho

Trace	Dispositivo	Tipo de rede	Número de Nós	Duração	Granularidade	Número de Contatos
Cambridge	*iMotes*	*Bluetooth*	54	11 dias	100 segundos	10873
Reality	*Phone*	*Bluetooth*	97	246 dias	300 segundos	54667

A seção a seguir apresenta de forma sucinta os algoritmos de roteamento utilizados neste trabalho.

4.2 Algoritmos de roteamento

Os algoritmos de roteamento utilizados na validação da proposta de gerenciamento do *buffer* foram o algoritmo Epidêmico e *PROpHET*. Esses algoritmos foram selecionados por dois motivos principais: primeiramente, tais algoritmos são baseados em replicações de mensagens, fato que aumenta a chance de *overflow* e assim beneficia a avaliação de políticas de gerenciamento de *buffer*. Segundo, estes algoritmos são bastante utilizados em propostas de novos algoritmos para o gerenciamento do *buffer* na literatura. Para maior entendimento do leitor, as subseções a seguir apresentam uma descrição dos algoritmos de roteamento utilizados neste trabalho.

4.2.1 Algoritmo Epidêmico

O algoritmo de roteamento Epidêmico é considerado a primeira proposta de roteamento para redes DTN, segundo Vahdat et al. [2000]. Esse protocolo considera que cada *host* da rede armazena uma tabela de *bits* denominada por *summary vector*. Basicamente, a ideia do algoritmo é comparar as tabelas de *bits* dos *hosts* envolvidos em uma oportunidade de contato. Realizada essa comparação, os nós requisitam entre si as mensagens que ainda não estão armazenadas no seu *buffer*. O objetivo principal do algoritmo é disseminar uma mensagem para todos os outros *hosts* da rede, com intuito de aumentar a probabilidade de entrega. A Figura 4.1 apresenta o funcionamento do algoritmo Epidêmico.

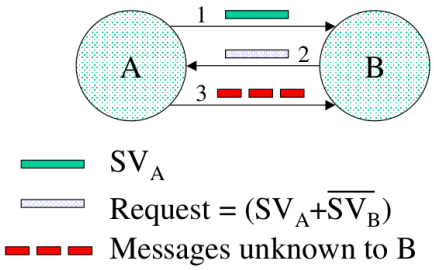

Figura 4.1. Funcionamento do algoritmo de roteamento Epidêmico quando dois nós A e B estão no mesmo raio de transmissão. (fonte: Vahdat et al. [2000])

A estratégia utilizada pelo algoritmo Epidêmico garante uma taxa de entrega próxima de 100%. No entanto, esses resultados somente são válidos se for considerado espaço infinito em *buffer*, o que em redes reais não se aplica. Segundo seus autores autores, os objetivos do algoritmo são maximizar a taxa de entrega, minimizar a latência e minimizar o total de recursos consumidos na rede. Apesar de maximizar de fato a taxa de entrega quando o tamanho do *buffer* é infinito, o algoritmo Epidêmico apresenta um desempenho bem aquém do esperado quando esse recurso é limitado. Ainda assim esse algoritmo é bastante utilizado como base na avaliação de novos protocolos de roteamento, gerenciamento de *buffer* e também de gerenciamento de energia para DTNs.

4.2.2 Algoritmo *PROpHET*

O algoritmo de roteamento *Probabilistic Routing Protocol using History of Encounters and Transitivity - PROpHET*, foi proposto em 2004 por Lindgren et al. [2003]. Esse algoritmo parte do princípio de que usuários não se movem aleatoriamente, como indica

o modelo de mobilidade *Random Waypoint* - RWP, mas sim de uma maneira previsível. Ou seja, se um nó visita várias vezes uma determinada localidade, existe uma chance muito grande de o mesmo visitar novamente tal localidade futuramente. Baseando-se nessa informação os autores criaram uma métrica de probabilidade chamada *delivery predictability*. Essa métrica indica a probabilidade de um nó entregar uma mensagem para outro nó destino.

Apesar de utilizar a probabilidade como métrica de escolha para qual mensagem rotear, o algoritmo *PROpHET* tem operação similar ao do algoritmo Epidêmico. Por exemplo, o algoritmo *PROpHET* utiliza uma estrutura denominada de *summary vector*, assim como o algoritmo Epidêmico. No entanto, diferente do algoritmo Epidêmico que guarda nesses vetores somente os índices das *mensagens* armazenadas no seu *buffer*, o algoritmo *PROpHET* armazena nesses vetores também informações de probabilidade de entrega de uma mensagem para os outros nós da rede. Quando ocorre o contato entre dois nós, os *summary vector* de cada *host* são trocados entre si. As informações de probabilidade de entrega são utilizadas por cada *host* para atualização interna das probabilidades e os índices das mensagens são utilizadas para requisitar mensagens de outros nós.

A escolha mais sofisticada do algoritmo *PROpHET* usando históricos de encontros dos nós e transitividade, apresentou resultados de desempenho superiores ao do algoritmo Epidêmico em ambientes baseados em comunidades. Em ambientes completamente aleatórios, o algoritmo *PROpHET* apresenta um desempenho parecido com o do algoritmo Epidêmico e algumas vezes até o supera. Dessa maneira esse algoritmo também é bastante utilizado na avaliação de novos protocolos para redes DTN.

4.3 Classificação dos laços sociais

Uma das etapas das simulações realizadas neste trabalho tratou da classificação da força do laço social entre os nós na rede. Este processo de classificação tornou-se necessário pelo fato de a política DLK considerar que são conhecidas a força das relações sociais entre os usuários da rede. Neste sentido, para cada trace utilizado no trabalho realizou-se um procedimento de classificação diferente, os quais são apresentados nas subseções a seguir.

4.3.1 Classificação da força das relações sociais no trace Reality

Para classificar-se a força da relação social entre dois nós no trace *Reality*, utilizou-se como base duas informações: quantidade de encontros entre os nós e histórico de ligações entre os participantes do experimento que gerou o trace, disponibilizado pelos idealizadores do mesmo. Baseando-se nessas duas informações, adotou-se os seguintes critérios para classificar-se a força dos laços sociais:

- Laço forte - é o laço entre dois nós que tem quantidade de encontros alto e que efetuaram chamadas entre si no histórico de ligações.
- Laço médio - é o laço entre dois nós que tem quantidade de encontros alto ou médio.
- Laço fraco - é o laço entre dois nós que não se encontram.

4.3.2 Classificação da força das relações sociais no trace de Cambridge

Diferentemente do trace *Reality*, no trace de *Cambridge* utilizou-se apenas a informação do número de encontros entre os nós para definir-se a força dos laços sociais entre os nós na rede. No entanto, utilizar somente esta informação poderia criar situações inconsistentes na classificação da força dos laços sociais. Por exemplo, se o único fator base para a classificação da força da relação social fosse o número de encontros, aqueles pares de nós que tivessem uma quantidade alta de encontros teriam um laço social forte. No entanto, no mundo real existem diversas situações onde isto não se aplica, por exemplo pessoas que se cruzam no caminho do trabalho diariamente. Apesar do alto número de encontros, essas pessoas a maioria das vezes não tem qualquer relação social. Além disso, existem pessoas que apesar de compartilharem de um laço social forte, tem uma quantidade de encontros muito baixa, geralmente por morarem distantes.

De acordo com essas informações, para representar tais situações utilizou-se o número de encontros como um dos fatores base para definir-se a *força da relação social* entre os nós da rede, mas distribuiu-se as relações sociais de acordo com a tabela 4.2.

Para as simulações realizadas com o trace de *Cambridge*, o processo de classificação da força dos laços sociais é composto por dois procedimentos básicos, que são a contagem da quantidade de encontros entre os nós e a classificação da força do laço em si.

Tabela 4.2. Critérios utilizados para classificação da força dos laços sociais entre os nós no trace de *Cambridge*

Tipo de laço	Distribuição	Quantidade de encontros
Laço Forte	15% dos nós	alta
Laço Forte	15% dos nós	baixa
Laço Médio	20% dos nós	média
Laço Médio	20% dos nós	baixa
Laço Fraco	20% dos nós	alta -
Laço Fraco	10% dos nós	alta -

Após executar-se o procedimento de contagem, para cada nó da rede é feita a classificação das força das relações com os demais nós, de acordo com os tipos de laços, a distribuição e a quantidade de encontros mostrados na tabela 4.1.

A distribuição apresentada na tabela 4.1 visa representar uma estimativa simplificada dentro do cenário de avaliação da quantidade de nós em cada tipo de força de laço social baseando-se no mundo real.

4.4 Métricas de avaliação

Segundo Fathima & Wahidabanu [2008], uma rede DTN pode ser avaliada por métricas de entrega de mensagens na rede. Dessa forma, selecionou-se na literatura para DTNs as métricas mais utilizadas na avaliação de desempenho em propostas de algoritmos para o gerenciamento do *buffer*. Assim, 3 métricas foram selecionadas, a saber:

- Taxa de entrega - Razão entre o número de mensagens recebidas e o número de mensagens enviadas.

- Atraso de Entrega - Intervalo de tempo médio entre o envio e o recebimento das mensagens na rede.

- Média de saltos - é a média de saltos de uma mensagem até ser entregue ao destino.

Essas métricas foram utilizadas nos experimentos realizados neste trabalho.

Os resultados obtidos em cada cenário para tais métricas são apresentados no próximo capítulo.

4.5 Políticas de gerenciamento de buffer utilizadas na avaliação de desempenho

Como citado anteriormente, além da política DLK, implementou-se nesse trabalho algumas políticas de gerenciamento do *buffer* disponíveis na literatura, com objetivo de obter-se uma visão qualitativa do desempenho da política de gerenciamento DLK. O ideal seria que fossem selecionadas para comparação outras políticas que considerassem aspectos sociais na gerência do *buffer*. No entanto, apesar de bastante atual, este tópico ainda não havia sido abordado com relação ao gerenciamento do *buffer* para DTNs.

Adotou-se então como critério para seleção das políticas, selecionar uma política de cada classe de políticas de gerenciamento de *buffer* existentes. Desta maneira, selecionou-se as seguintes políticas:

- *Drop Oldest* - Política pertencente a classe de políticas que utilizam informações locais. Realiza o descarte da mensagem mais antiga no *buffer*
- *Drop least recently received* - LRR - Política pertencente a classe de políticas que utilizam informações locais. Realiza o descarte da mensagem menos recentemente recebida
- *Evict most forwarded first* - MOFO - Política pertencente a classe de políticas que utilizam informações parciais do estado da rede. Realiza o descarte das mensagens enviadas um número máximo de vezes

4.6 Parâmetros gerais das simulações

Alguns parâmetros das simulações foram utilizados como descrito na tabela abaixo.

Tabela 4.3. Parâmetros utilizados nos experimentos

Velocidade de Transmissão	250 kBps
Quantidade de Mensagens	1 mensagens/min
Tamanho das mensagens	500 kb a 1 MB
TTL	5 horas

Para aumentar-se a confiabilidade nos resultados obtidos com a avaliação de desempenho da política DLK, utilizou-se nas simulações um ferramental estatístico para o tratamento dos dados. As subseções a seguir apresentam detalhes sobre os métodos estatísticos utilizados neste trabalho.

4.7 Métodos Estatísticos

Para avaliação de desempenho da política DLK, realizou-se simulações de horizonte infinito. A justificativa é muito simples: não é possível conhecer antecipadamente o tamanho da amostra que vai produzir a resposta de uma simulação com uma precisão relativa desejada. A única maneira de controlar o erro estatístico de um procedimento de estimação é analisar os dados sequencialmente Mota [2002]. Neste trabalho, a largura do intervalo de confiança foi obtida implementando-se uma versão sequencial do método *Overlapping Batch Means* (OBM) Meketon & Schmeiser [1984], que processa um conjunto de observações da métrica de desempenho de interesse, no estado de equilíbrio dinâmico (estado estacionário). Para alcançar esse estado, implementou-se o método *Mean-Squared-Error-Reduction* (MSER-5) Franklin & White [2008] para detectar o fim do período transiente e descartar as observações coletadas até então.

Após seguir a mesma heurística do método clássico *Nonoverlapping Batch Means*, onde é encontrado M, o tamanho ótimo de B blocos contíguos de observações cujas médias podem ser consideradas sem correlação dentro de um certo nível de significância desejado, o OBM agrupa as observações do estado estacionário em blocos de tamanho M. O bloco 1 começa na observação 1 vai até a observação M; o segundo bloco começa na observação 2 e vai até a observação M+1, e assim por diante. Portanto, obtém-se N - M + 1 blocos. As médias desses blocos são obtidas, assim como a média global e a variância, informações necessárias para se construir um intervalo de confiança de 95%. Se o erro relativo pretendido (5%), definido como a relação entre a metade da largura do intervalo de confiança e a média global, não foi alcançado, aumenta-se o valor de M e a simulação continua coletando mais observações até obter o valor de N igual a B.M, quando o teste do erro relativo citado acima é feito mais uma vez. A simulação termina quando a precisão relativa desejada é alcançada.

4.7.1 Método MSER-5

O fim do estado transiente ou ainda ponto de truncagem, deve ser aquele a melhorar a qualidade da função de aproximação de uma simulação em menos tempo com o menor impacto possível na precisão dos dados. A utilização do método MSER-M Franklin & White [2008] apresenta-se com um dos melhores métodos para esta análise.

O método MSER-M consiste em encontrar o ponto ótimo de truncagem em uma séria de amostras g divididas em *batches* (lotes) de tamanho m, ou seja, este método é aplicado sobre uma série de lotes cujos elementos correspondem a média de m elementos consecutivos da série.

O MSER-5 é uma variação do MSER-m quando m é igual a 5. Esta alteração apresentou melhores resultados segundo os testes realizados em Franklin & White [2008].

4.7.2 Método Overlapping Batch Means

Com o objetivo de estimar variância dos valores obtidos em um processo estocástico, foi apresentado em 1984 por Meketon & Schmeiser [1984] o método *Overlapping Batch Means* - OBM, uma variação do método *Nonoverlapping batch means* - NOBM, porém com 2/3 da variância deste.

Nesse método, primeiramente deve-se dividir as amostras em B blocos, cada um com tamanho M. Em seguida, coletam-se observações até que seja possível preencher todos os blocos. Posteriormente, calculam-se as médias de cada bloco, para que seja possível verificar se a nova amostra gerada pelas médias dos blocos apresenta correlação e, caso apresente, utiliza-se um teste para tornar a amostra livre de correlação. Esta nova organização gera um total de N * M + 1 e não mais B blocos. Uma vez com os blocos organizados, calcula-se a média dos blocos (X) e o tamanho relativo do intervalo de confiança (H). Se o tamanho do intervalo de confiança for maior ou igual ao pretendido, a simulação é finalizada, senão, o tamanho dos blocos deve ser aumentado e deve-se coletar mais amostras até que N = B * M, para, em seguida, continuar-se verificando até que a condição seja satisfeita.

4.7.3 Correlação de Von Neuman

Em teoria de probabilidade e estatística, correlação, ou coeficiente de correlação, indica a força e a direção do relacionamento linear entre as variáveis aleatórias.

Para realização deste teste, utilizou-se a correlação de *von Neuman*, que a partir de uma sequência de B médias de blocos, verifica se os dados ainda apresentam correlação. Para o cálculo desta medida, utiliza-se:

$$RVN = \frac{\sum j = 1^{B-1}(Rj - Rj + 1)^2}{\sum j = 1^{B}(Rj - R)^2} \tag{4.1}$$

onde,

$$R = \frac{\sum k = 1^{B} Rk}{B} \tag{4.2}$$

Se o resultado deste cálculo for maior que o valor crítico da tabela de *von Neuman*, indica que as médias dos blocos ainda possuem correlação, logo essa sequência não

poderia ser usada na simulação. A solução seria aumentar o tamanho do bloco e refazer o cálculo até que a correlação entre os valores seja aceitável.

A forma como foram implementados os métodos e técnicas utilizadas neste trabalho no simulador One, são descritos no apêndice A.

Capítulo 5

Resultados

Como citado na seção 4.4 do capítulo anterior, para efeito de comparação, foram selecionadas algumas políticas disponíveis na literatura para DTNs. As políticas selecionadas foram *Drop Oldest*, *Drop Last Recent Received* - LRR e *Evict Most Forwarded First* - MOFO.

As simulações foram realizadas variando-se o tamanho do *buffer* de 10 MB a 80 MB. A seguir são apresentados os resultados para as três métricas avaliadas neste trabalho em cada um dos conjuntos de dados utilizado.

5.1 Resultados dos Experimentos usando algoritmo Epidêmico

5.1.1 Taxa de Entrega

A taxa de entrega é definida como a razão entre a quantidade de mensagens entregues ao destinatário e a quantidade de mensagens criadas. Neste trabalho, analisou-se a taxa de entrega de duas formas. Primeiramente, foi realizada uma comparação dos resultados obtidos por todas as políticas implementadas. Segundo, analisou-se a taxa de entrega por quantidade de mensagens entregues por tipo de força do laço social. As equações 5.1 e 5.2 denotam as fórmulas utilizadas nas duas comparações realizadas.

$$TaxadeEntrega = \frac{qtde.mensagensentregues}{qtde.mensagenscriadas} \tag{5.1}$$

$$TaxaPorForadoLaoSocial = \frac{qtde.mensagensentreguesporclasse}{qtde.mensagenscriadasporclasse} \tag{5.2}$$

A comparação do desempenho de DLK com o das demais política com relação à taxa de entrega no cenário *Cambridge*, são apresentados nas Figuras 5.1 e 5.2. Para ambos os casos, os resultados obtidos mostram que a política DLK aumenta a taxa de entrega nos cenários avaliados.

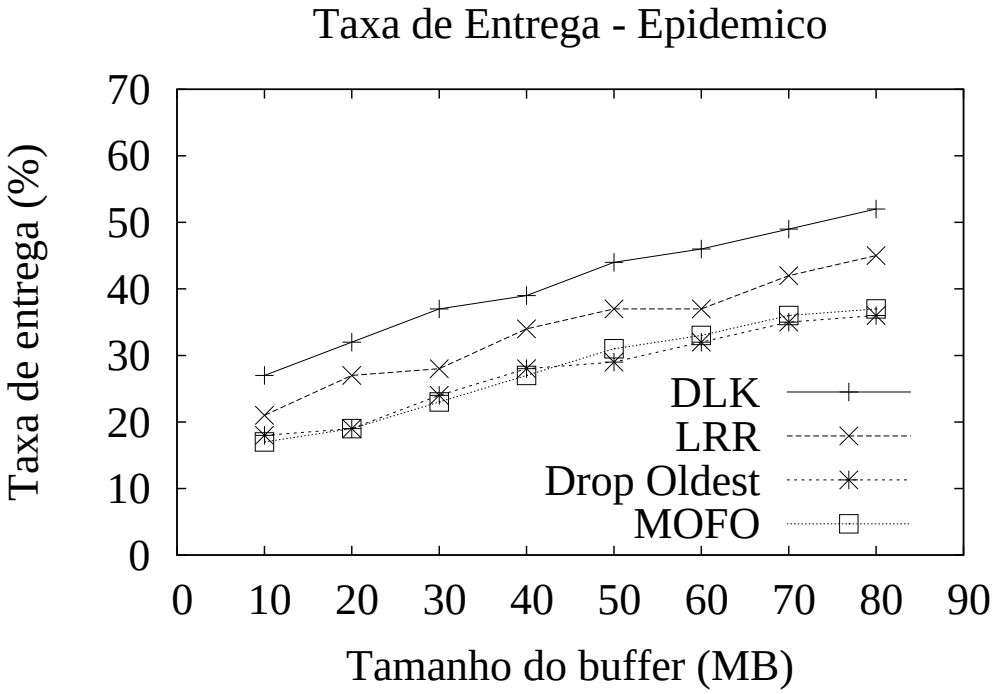

Figura 5.1. Taxa de entrega utilizando roteamento Epidêmico no trace de *Cambridge*

No trace de *Cambridge*, a política DLK superou as demais políticas avaliadas. É interessante notar que a diferença a diferença do desempenho de DLK para a segunda colocada manteve-se entre 5 e 7 pontos percentuais em quase todos os tamanhos de *buffer* testados. Pode-se destacar também o desempenho das políticas MOFO e *DO* que foram parecidos e em alguns pontos até mesmo igual. Por exemplo, para tamanho de *buffer* igual a 20 MB, onde política LRR obteve o segundo melhor desempenho. ambas as políticas entregaram 19% das mensagens criadas. A maior diferença no resultado obtido por estas políticas foi de 4 pontos percentuais, para tamanho de *buffer* igual a 50 MB. A política LRR obteve o segundo melhor resultado, se aproximando em até 5 pontos percentuais do desempenho da política DLK.

A Figura 5.2 apresenta o resultado da taxa de entrega no cenário *Reality*. Pode-se perceber que a política DLK obteve a melhor taxa de entrega no cenário avaliado superando as demais políticas testadas. A política DO obteve o segundo melhor resultado, mantendo um desempenho inferior ao de DLK variando entre 3 e 7 pontos

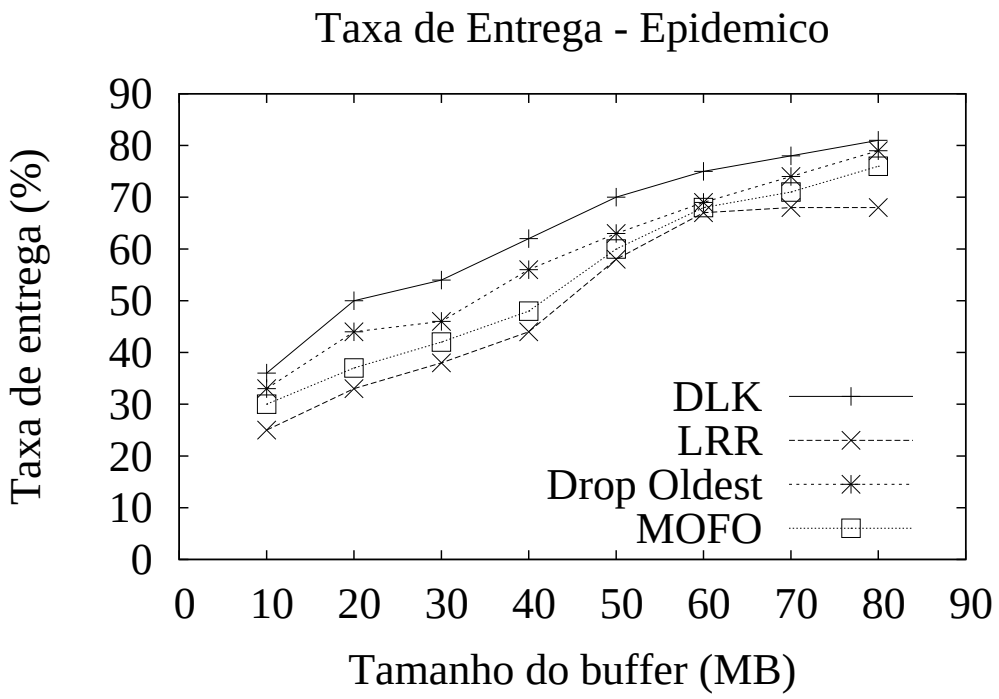

Figura 5.2. Taxa de entrega utilizando roteamento Epidêmico no trace *Reality*

percentuais. É interessante observar que todas as políticas obtiveram um comportamento semelhante: quanto maior o tamanho do espaço em *buffer*, maior a taxa de entrega. A justificativa para este fato é simples: quanto mais espaço, mais mensagens podem ser armazenadas e, consequentemente, mais mensagens poderão ser enviadas em uma oportunidade de contato, fato que aumenta a probabilidade de entrega. As políticas LRR e MOFO obtiveram um resultado parecido. É interessante destacar que para tamanho de *buffer* igual a 60 MB, as políticas LRR, DO e MOFO obtiveram um desempenho praticamente igual e em outros pontos bastante próximo, como por exemplo para *buffer* igual a 50 e 70 MB. No entanto em ambas as situações as diferenças variaram entre 1 e 3 pontos percentuais.

No caso da combinação de roteamento Epidêmico com gerenciamento de *buffer* feito por DLK, pode-se afirmar que a política DLK serve como uma poda das mensagens que serão armazenadas no *buffer*. As replicações de várias cópias de uma mesma mensagem realizada pelo algoritmo Epidêmico, fazem com que diversas mensagens endereçadas a nós poucos conhecidos por um determinado nó sejam repassadas a ele. Dessa forma, a política DLK descarta primeiramente, em casos de *overflow*, estas mensagens, priorizando a manutenção em *buffer* das mensagens destinadas aos nós com quem se mantém um laço social mais forte, aumentando assim a probabilidade

desta mensagem ser entrega, fato corroborado pelo desempenho apresentado nas Figura 5.1 e 5.2.

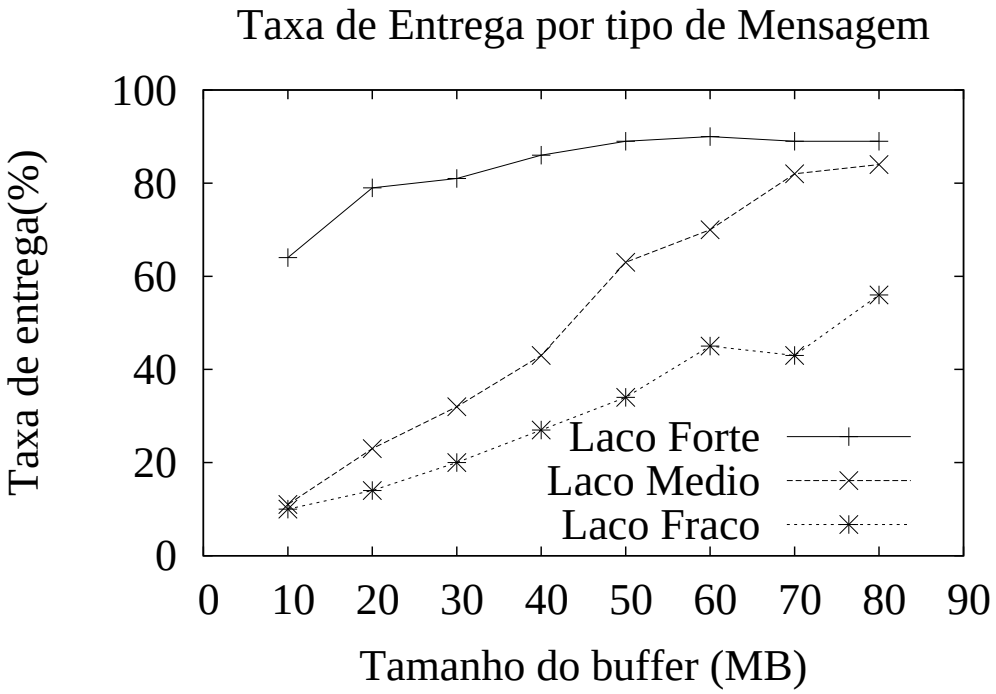

Figura 5.3. Taxa de entrega de acordo com a força do laço social no trace *Reality*

As Figuras 5.3 e 5.4 apresentam a taxa de entrega de acordo com o tipo da mensagem, como indicado na equação 5.2. Pode-se notar no gráfico que as mensagens para usuários com quem se tem um laço mais forte são entregues em taxas bem maiores do que para os outros tipos de laços. No cenário *Reality*, até 88% das mensagens destinadas a nós com quem o laço é forte, foram entregues. No cenário *Cambridge*, essa taxa chegou a até 76%. Para tamanhos menores de *buffer* a diferença entre as taxas de mensagens entregues de acordo com tipo do laço, chegou a até aproximadamente 45 pontos percentuais no trace *Reality* e de 40 no trace de *Cambridge*.

Com estes resultados, pode-se observar a melhoria na entrega de mensagens na rede, fornecida pela política DLK. A estratégia adotada pela política DLK beneficia de forma natural o aumento da taxa de entrega na rede, visto que sempre são mantidas no *buffer* mensagens que o usuário, por algum motivo, gostaria de carregar e provavelmente entregará ao destinatário. Este fato é corroborado pelos resultados exibidos nos gráficos 5.1 e 5.2. No entanto, ao mesmo tempo que beneficia a taxa de entrega, DLK pode aumentar o atraso médio de entrega caso o tempo médio de encontro com os usuários que deseja-se ajudar sejam altos.

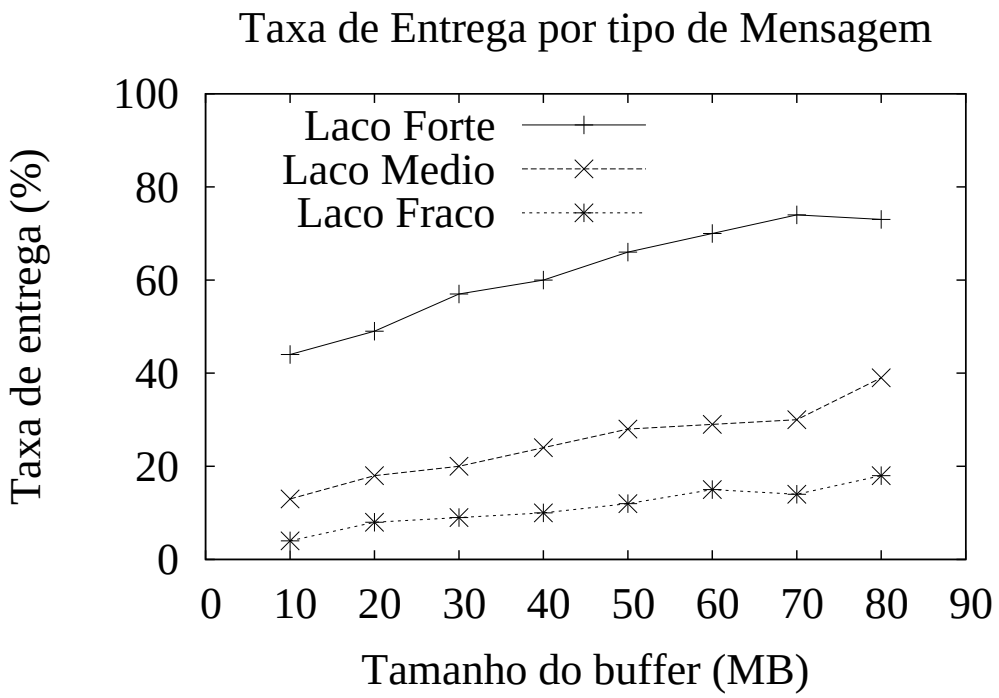

Figura 5.4. Taxa de entrega de acordo com a força do laço social no trace *Cambridge*

5.1.2 Atraso Médio de entrega

O atraso médio de entrega é definido como o intervalo de tempo médio entre o envio e o recebimento das mensagens na rede. Desta forma, é importante que o atraso médio de entrega em uma DTN seja minimizado. Neste trabaho, o atraso médio de entrega foi avaliado comparando-se o desempenho da política DLK com o das demais políticas implementadas para comparação. Os resultados apresentados nos gráficos 5.5 e 5.6, apresentam os resultados obtidos nos experimentos realizados, com relação à métrica atraso médio de entrega, nos cenários *Cambridge* e *Reality*. Pelo fato do contato entre dois nós em DTNs ocorrer em períodos que podem ser variados, o atraso na comunicação pode ser da ordem de horas e até mesmo chegar a dias, fato corroborado pelos resultados obtidos.

A Figura 5.5 apresenta os resultados obtidos no cenário *Reality*. Nota-se que a política DLK obteve o menor atraso médio de entrega neste cenário. No entanto, todas as políticas obtiveram um comportamente semelhante: quanto maior o espaço em *buffer*, menor o atraso médio de entrega obtido. A política DO obteve o segundo melhor resultado, superando as políticas MOFO e LRR.

A Figura 5.6 apresenta os resultados para a métrica atraso médio de entrega no

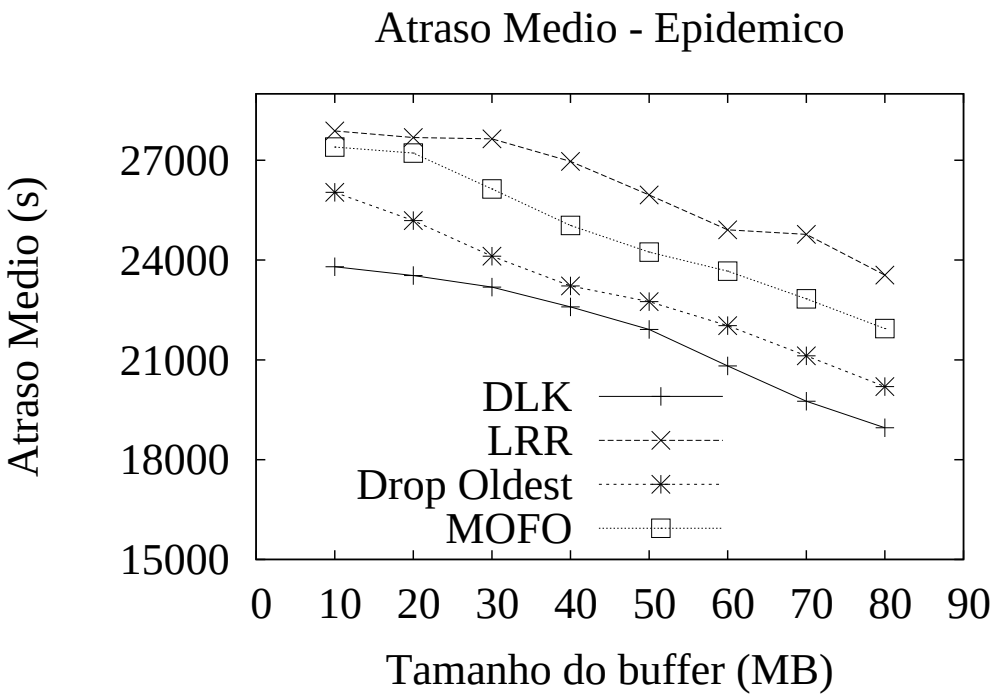

Figura 5.5. Atraso Médio utilizando roteamento Epidêmico no trace de *Reality*

cenário *Cambridge*. Novamente a política DLK obteve o menor atraso na entrega das mensagens. A política DO obteve o segundo melhor resultado, novamente, e as políticas MOFO e LRR apenas alteraram as suas colocações com relação aos resultados obtidos no cenário *Reality*. No cenário *Cambridge* a política MOFO ficou a frente da política LRR, com exceção para tamanho de *buffer* igual a 80 MB, enquanto que no cenário *Reality* o inverso ocorreu. Pode-se destacar também o desempenho bastante parecido obtido pela política DLK para a maioria dos tamanhos de *buffer* testados. Percebe-se um comportamento praticamente linear até tamanho de *buffer* igual a 60 MB. A partir de tamanho de *buffer* igual a 70 MB, o desempenho da política DLK aumentou ao entregar as mensagens mais rapidamente, fato que diminuiu o atraso médio de entrega.

Os resultados obtidos com relação a métrica atraso médio de entrega nos cenários avaliados neste trabalho, mostram que a política DLK pode diminuir o atraso na entrega de mensagens na rede. No entanto, acredita-se que o desempenho da política DLK com relação a esta métrica seja influenciado pela frequência com que os nós se encontram dentro da rede, pois o contato entre os nós ajudam a entrega de uma mensagem em um menor tempo, consequentemente, diminuindo o atraso na entrega. Nesse caso, aumentando-se a frequência com que os nós que mantém uma relação social forte se encontram na rede, consequentemente diminui-se o tempo necessário para repassar

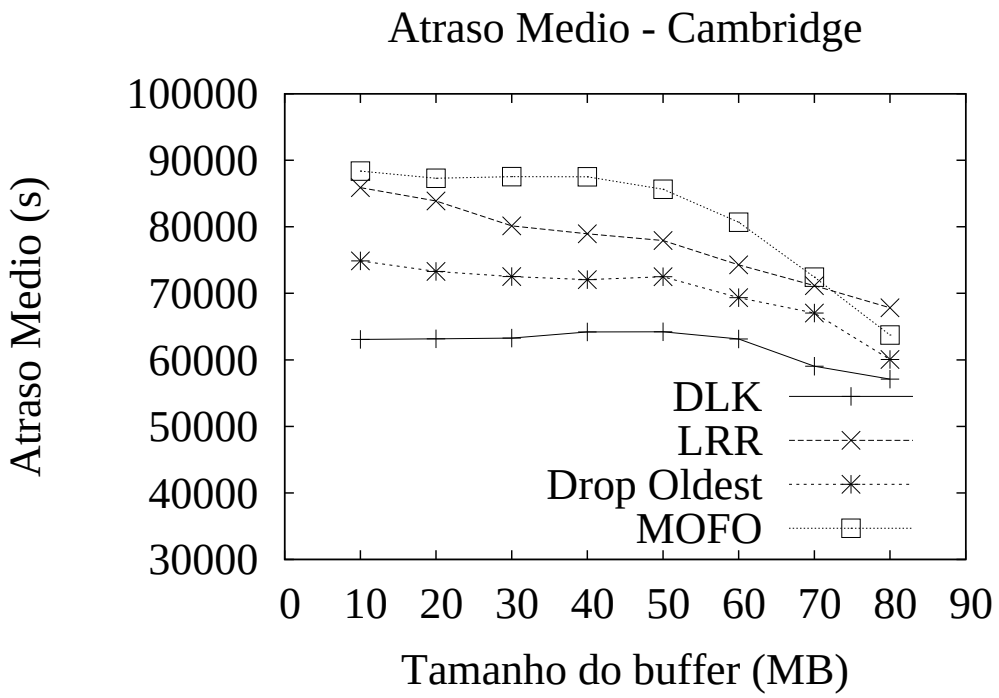

Figura 5.6. Atraso Médio utilizando roteamento Epidêmico no trace de *Cambridge*

uma mensagem ao destino, o que contribui para que a política DLK diminua o atraso médio, fato corroborado nas Figuras 5.5 e 5.6. Entretanto, caso a frequência de encontros entre os nós seja diminuída, a política DLK preservará a probabilidade de entrega, porém aumentará o tempo necessário para entregar a mensagem ao destino e assim aumentará o atraso médio.

5.1.3 Média de Saltos

A média de saltos é definida como a quantidade média de saltos de uma mensagem até ser entregue ao destino. Neste trabalho, avaliou-se a métrica média de saltos de duas formas. Primeiramente, foi realizada uma comparação dos resultados obtidos por todas as políticas implementadas. Segundo, analisou-se a média de saltos para as mensagens entregues de acordo com o tipo de força do laço social. As equações 5.3 e 5.4 denotam as fórmulas utilizadas nas duas comparações realizadas.

$$MediadeSaltos = \frac{TotaldeSaltosporMensagemEntregue}{TotalMensagensEntregues} \tag{5.3}$$

$$MediadeSaltosporTipodeMensagem = \frac{TotaldeSaltosporTipodeMensagemEntregues}{TotaldeMensagensEntreguesporTipo} \quad (5.4)$$

As Figuras 5.7 e 5.8 apresentam os resultados obtidos para a métrica média de saltos nos cenários *Cambridge* e *Reality*. Pode-se notar a partir dos resultados obtidos que, quanto maior o espaço disponível de *buffer*, menor é a quantidade média de saltos para a maioria das políticas. No entanto, a política DLK tende a se manter estável, obtendo um leve crescimento devido a maior disponibilidade de recursos para manter as mensagens por maior tempo na rede, dando assim uma maior probabilidade de entrega para as mensagens armazenadas.

Para as demais políticas avaliadas, a média de saltos possui um comportamento descendente em ambos os cenários avaliados, ocasionado pela alta taxa de replicação de mensagens na rede, realizada pelo algoritmo Epidêmico que, com mais recursos à disposição, consegue manter o conteúdo por mais tempo no *buffer*, diminuindo o descarte de mensagens e, consequentemente, a quantidade de saltos necessária para enviar para o destino uma determinada mensagem.

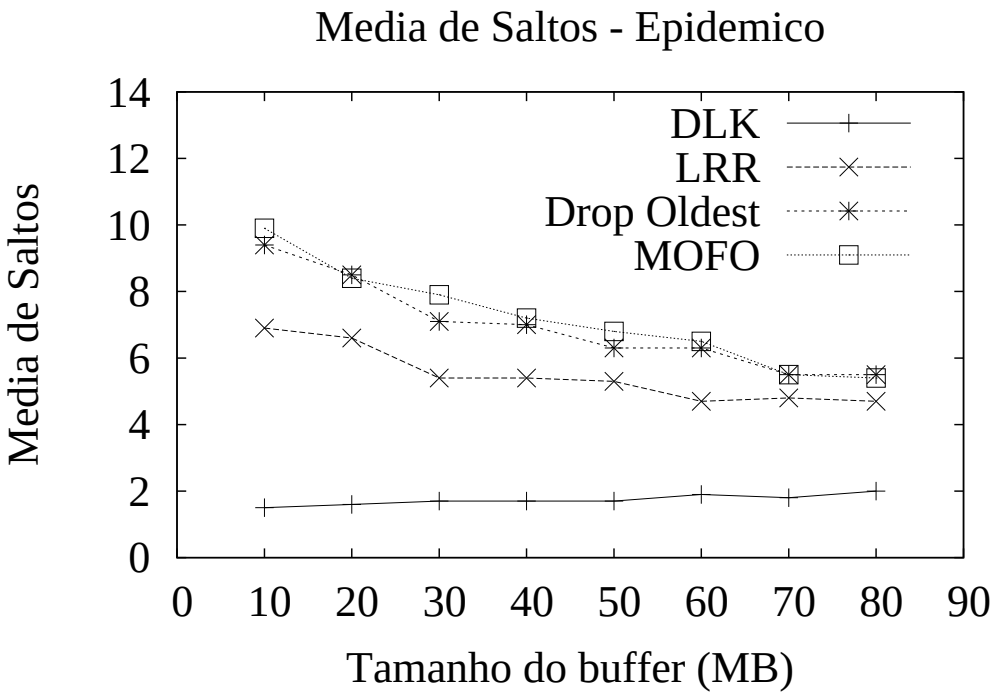

Figura 5.7. Média de saltos utilizando roteamento Epidêmico no trace de *Reality*

Na Figura 5.7 e na Figura 5.8, percebe-se que a política DLK obteve a menor

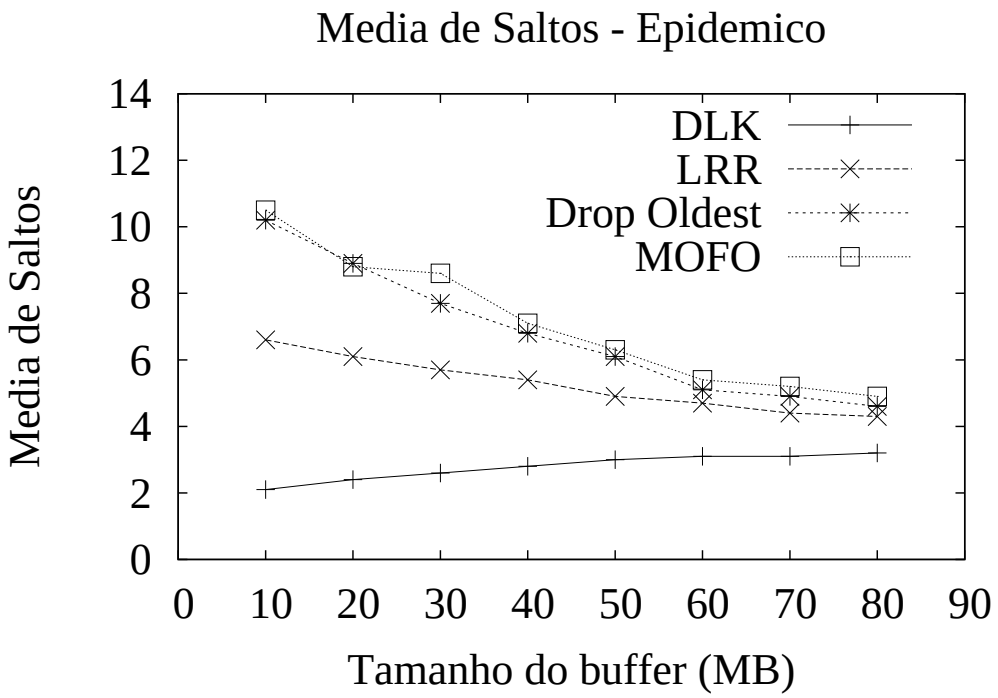

Figura 5.8. Média de saltos utilizando roteamento Epidêmico no trace de *Cambridge*

média de saltos, tendo como maior média 3 quantidades de saltos necessários para uma mensagem chegar ao destino final. É interessante notar que as políticas DO e MOFO apresentam um desempenho bem parecido. Isso se deve as características das duas políticas, que tem por um dos objetivos manter a mensagem na rede o máximo de tempo possível, com objetivo de prover uma melhor taxa de entrega. Nota-se também que a política LRR tem uma média de saltos menor e com uma fração descendente, causado pela característica da política LRR que favorece eventualmente contatos mais frequentes entre pares de nós.

As Tabelas 5.1 e 5.2 apresentam a média de saltos para mensagens por tipo de laço social em cada um dos cenários avaliados neste trabalho. Pode-se perceber que a variação do tamanho em *buffer* influenciou para o aumento da média de saltos necessárias para uma mensagem ser entregue ao destino. Isto se deve ao fato de que um espaço maior em *buffer* favorece o armazenamento de mais mensagens, o qual, ao utilizar-se o algoritmo de roteamento Epidêmico, possibilita que uma quantidade maior de mensagens sejam transmitidas em uma oportunidade de contato. Desta maneira, como a duração de um contato em DTNs é variável, quando se tem mais mensagens para se transmitir em uma oportunidade de contato, pode ocorrer a desconexão, fato que impedirá que todas as mensagens sejam transmitidas em uma oportunidade de contato,

podendo aumentar assim a quantidade média de saltos necessária. É interessante notar também que, em ambos os resultados obtidos nas Tabelas 5.2 e 5.3, a quantidade média de saltos necessárias para uma mensagem do tipo de laço social mais forte ser entregue é menor que para as mensagens de outros tipos de laços.

Tabela 5.1. Média de saltos apresentado pela política DLK no trace *Cambridge*

Tipo	10 MB	20 MB	30 MB	40 MB	50 MB	60 MB	70 MB	80 MB
Laço Forte	1,2	1,4	1,5	1,7	1,7	1,8	1,8	2
Laço Médio	1,3	1,5	1,7	1,8	1,9	2,2	2,1	2,1
Laço Fraco	1,4	1,7	1,8	1,9	2	2,5	2,3	2,6

Tabela 5.2. Média de saltos apresentado pela política DLK no trace *Reality*

Tipo	10 MB	20 MB	30 MB	40 MB	50 MB	60 MB	70 MB	80 MB
Laço Forte	1,3	1,4	1,8	2,2	2,1	2,5	2,3	2,7
Laço Médio	2,3	2,5	2,5	2,5	2,7	2,9	3,1	3,1
Laço Fraco	2,4	2,4	2.8	3,1	3,4	3,5	3,6	3,6

Com os resultados obtidos nos cenários valiados com relação à métrica média de saltos, pode-se concluir que a política DLK pode contribuir para uma diminuição da quantidade média de saltos necessária para que uma mensagem seja entregue. No entanto, é interessante destacar que o bom desempenho da política DLK com relação à métrica em questão, não depende da taxa de encontros entre os nós na rede, diferentemente do caso da métrica atraso média de entrega. Ou seja, a taxa de encontros não influência o desempenho de DLK com relação à métrica média de saltos.

5.2 Resultados dos Experimentos usando algoritmo de roteamento *PROpHET*

Na validação do algoritmo de gerência de *buffer* DLK, foram também realizados experimentos com o algoritmo de roteamento PROPhET, visando avaliar métricas de entrega de mensagens na rede. A seguir são apresentados os resultados obtidos para as três métricas observadas neste trabalho.

5.2.1 Taxa de entrega

Da mesma maneira que com o algoritmo Epidêmico, a taxa de entrega também foi avaliada de duas maneiras, para a combinação do algoritmo PROPhET com a política DLK. Primeiramente é realizada uma comparação do desempenho apresentado

pela combinação do algoritmo de roteamento PROPhET com as políticas implementadas neste trabalho. Segundo, realizou-se uma comparação da quantidade de mensagens entregues por tipo de força de laço social individualmente. As Fórmulas 5.1 e 5.2 representam as equações utilizadas em cada uma destas maneiras de avaliação. As Figuras 5.9 e 5.10 apresentam os desempenhos obtidos nos cenários *Cambridge* e *Reality*.

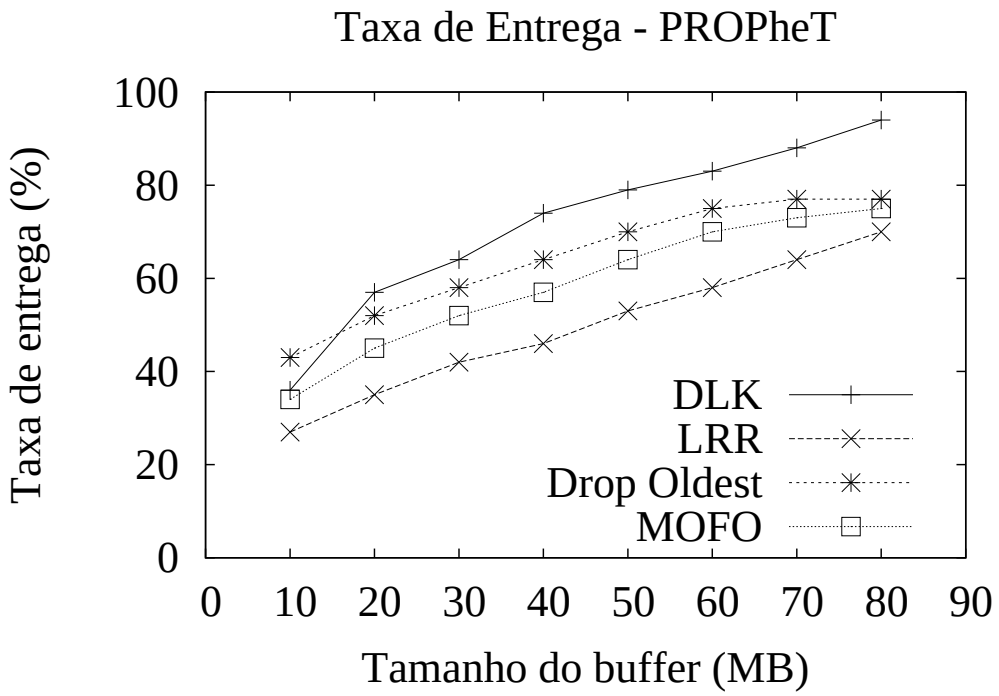

Figura 5.9. Taxa de entrega utilizando roteamento *PROpHET* no trace de *Reality*

A Figura 5.9 apresenta o resultado obtido no cenário *Reality*. Pode-se perceber que a política DLK obteve o melhor desempenho, superando as suas oponentes. É interessante notar a taxa de entrega para o tamanho de *buffer* igual a 80 MB, quando a política DLK entregou até 93% das mensagens criadas. Este fato pode ser justificado pela combinação da utilização de probabilidade do algoritmo PROpHET e a poda feita pela política DLK, a qual mantém no *buffer* mensagens destinadas aos nós com quem se mantém maior relação social, fato que aumenta a chance de uma mensagem ser entregue. A política DO obteve o segundo melhor resultado, superando as políticas LRR e MOFO. É interessante observar que, de fato, a política DO favorece o funcionamento do algoritmo PROpHET, visto que mantém uma mensagem o máximo de tempo possível na rede, tornando-a disponível para seu encaminhamento pelo algoritmo PROpHET. As políticas MOFO e LRR obtiveram os piores desempenhos com

relação à taxa de entrega no cenário em questão, mantendo-se uma diferença de 5 a 10 pontos percentuais com vantagem para a política MOFO.

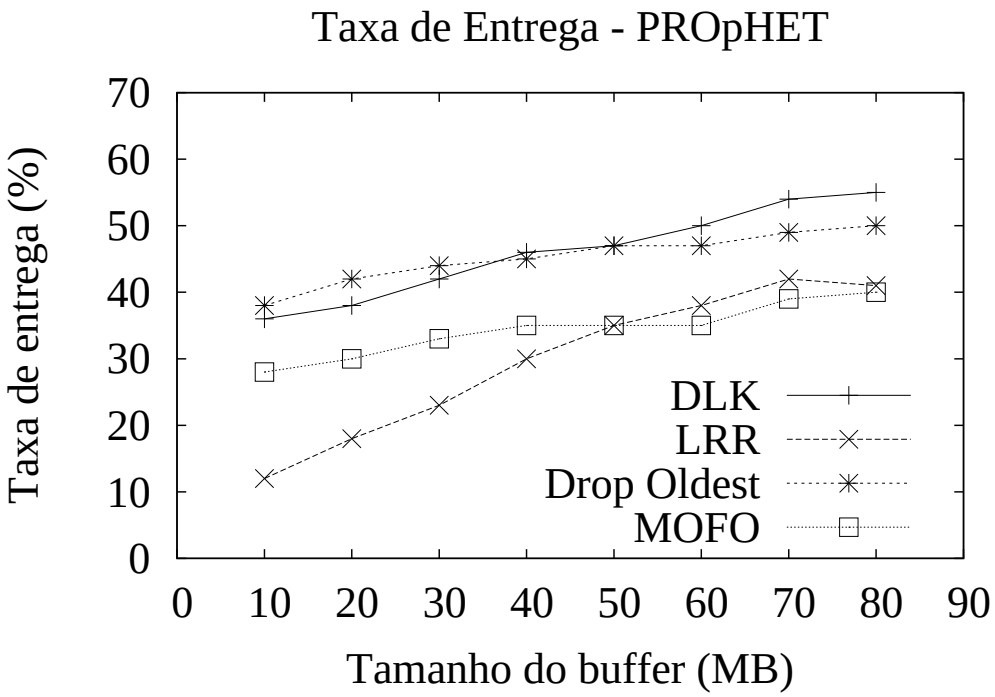

Figura 5.10. Taxa de entrega utilizando roteamento *PROpHET* no trace de *Cambridge*

A Figura 5.10, por sua vez, apresenta os resultados para a taxa de entrega obtida no cenário de *Cambridge*. Neste cenário, a política DLK superou as demais políticas testadas para tamanhos de *buffer* iguais a 40, 60, 70 e 80 MB. Para os demais tamanhos de *buffer*, a política *Drop Oldest* obteve o melhor resultado. As políticas MOFO e LRR, por sua vez, obtiveram desempenho parecido para tamanhos de *buffer* maiores que 40 MB, mas abaixo das demais oponentes, em uma diferença que variou entre 10 e 12 pontos percentuais.

A Figura 5.11 apresenta os resultados no trace *Reality* da razão dada pela Fórmula 5.2. É interessante notar que, para determinados tamanhos de *buffer*, aproximadamente 90% das mensagens destinadas a usuários com quem se tem um laço forte, foram entregues. Essa taxa é superior aos demais tipos de laços considerados neste trabalho, chegando a diferenças de até 40 pontos percentuais de mensagens a mais sendo entregues.

A Figura 5.12, por sua vez, apresenta a mesma comparação dada pela Fórmula 5.2, mas no cenário *Cambridge*. Novamente, a quantidade de mensagens entregues de laços mais fortes foi superior aos demais tipos de laços. A diferença entre a quantidade

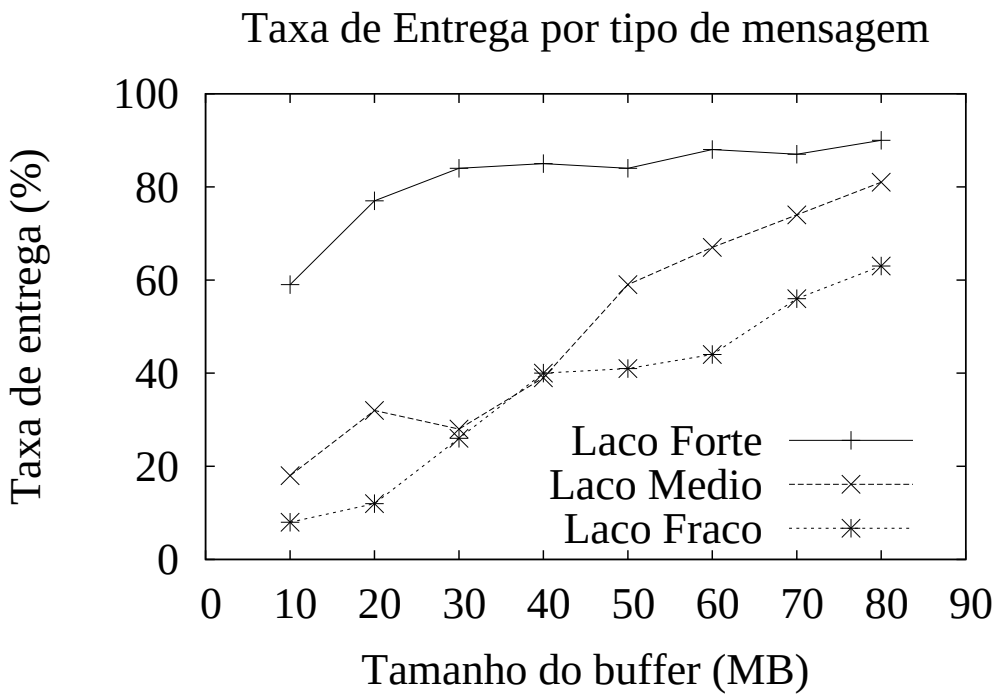

Figura 5.11. Taxa de entrega de acordo com a força do laço social no trace *Reality*

de mensagens entregues por tipo de laço foi de até 40 pontos percentuais de vantagem para as mensagens destinadas a nós com quem se tem um laço mais forte. É interessante observar que, no trace de *Cambridge*, a quantidade de mensagens do tipo de laço médio foi menor com relação ao trace *Reality*. Esta diferença foi de até 40%, visto que no trace *Reality*, para tamanho de *buffer* igual a 80 MB, 80% das mensagens destinadas a nós do tipo de laço médio foram entregues.

Da mesma maneira que nos experimentos com o algoritmo PROPhET, os resultados apresentados nas Figura 5.11 e 5.12, podem ser justificados pela característica da política DLK, a qual realiza uma poda no *buffer* dos nós, mantendo sempre prioritariamente mensagens destinadas a usuários com quem se tem um laço social mais forte. Dessa maneira, quanto mais mensagens dos laços mais fracos forem descartadas, menor será a chance de ocorrer sucesso na entrega das mensagens desse tipo, fato corroborado pelos resultados apresentados nas Figuras 5.11 e 5.12.

Através destes resultados, pode-se afirmar que, a probabilidade baseada em encontros utilizada pelo algoritmo *PROpHET* aliada á escolha baseada na força dos laços sociais entre os nós realizada pela política DLK, contribuiu para que o desempenho obtido por esta combinação seja superior ao das demais políticas testadas em ambos os cenários utilizados neste trabalho, alcançando, em alguns casos, taxas de até 90% das

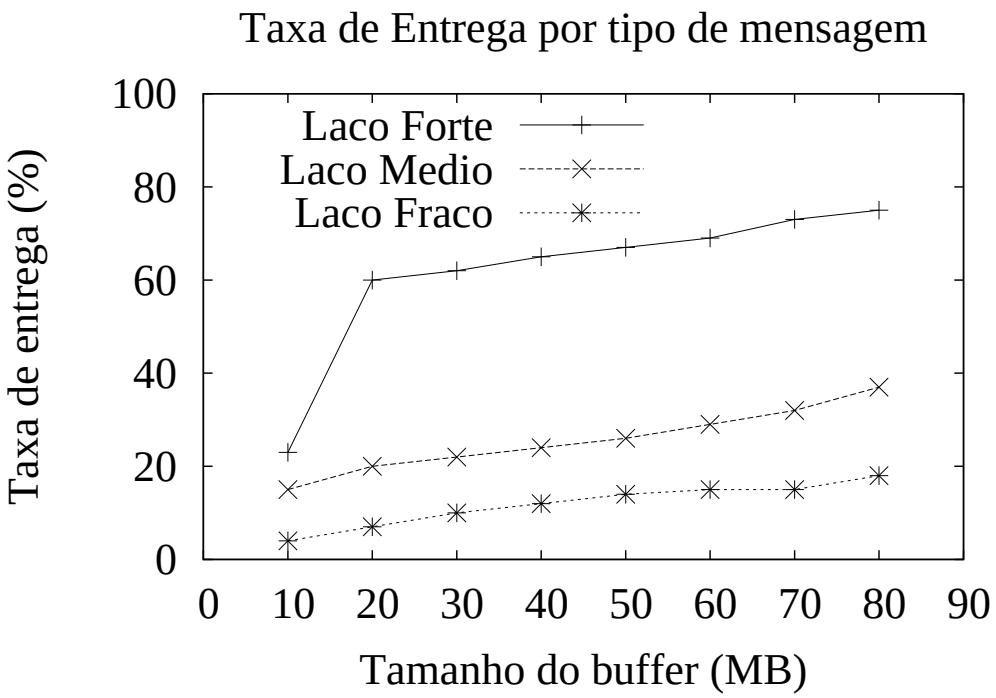

Figura 5.12. Taxa de entrega de acordo com a força do laço social no trace de *Cambridge*

mensagens entregues.

5.2.2 Atraso Médio

Como citado anteriormente, o atraso médio é definido como o intervalo de tempo médio entre o envio e o recebimento das mensagens na rede. As Figuras 5.13 e 5.14 apresentam os resultados obtidos com relação a métrica atraso médio de entrega nos cenários de *Cambridge* e *Reality*.

A Figura 5.13 apresenta o resultado do atraso médio de entrega no cenário de *Cambridge*. Observa-se uma diferença considerável no desempenho das políticas DLK e LRR com relação as políticas DO e MOFO. A política LRR obteve o desempenho mais próximo de DLK mantendo-se em uma diferença aparentemente constante e, para tamanho de *buffer* igual a 80 MB, LRR obteve um atraso médio de entrega menor que DLK. As políticas DO e MOFO obtiveram os atrasos médios mais altos. A diferença no desempenho entre essas políticas também se manteve aparentemente constante para tamanhos menores de *buffer*, sofrendo um leve aumento para tamanhos de *buffer* maior. É interessante destacar que para todas as políticas o comportamento foi o mesmo: quanto maior o espaço em *buffer*, menor o atraso médio de entrega.

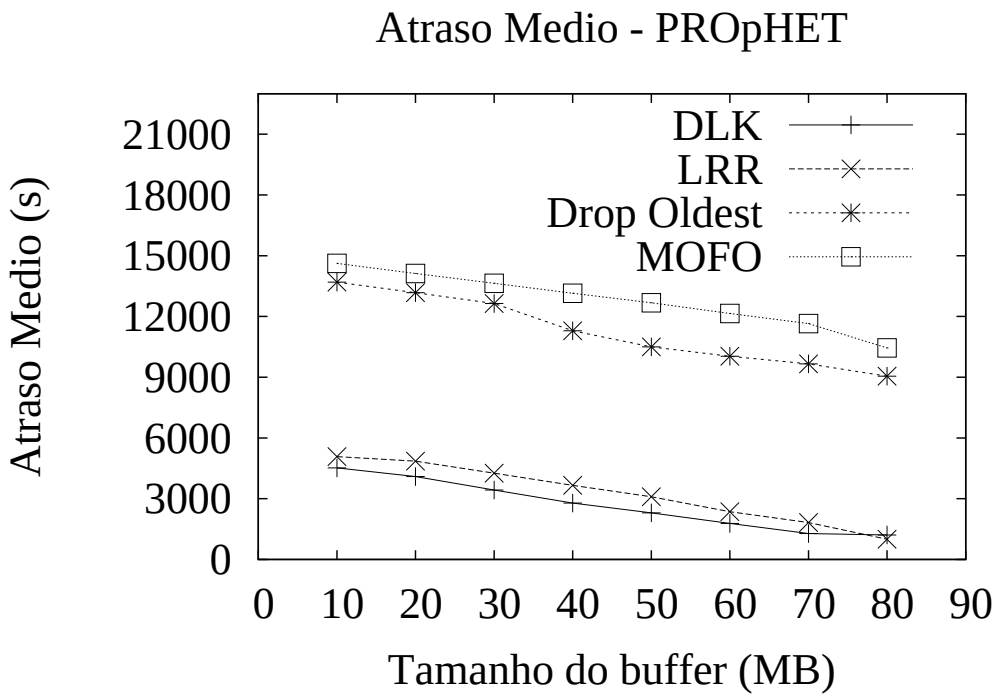

Figura 5.13. Atraso Médio utilizando roteamento *PROpHET* no trace de *Cambridge*

Um fato interessante que observa-se está no atraso médio mínimo e máximo no trace de *Cambridge* quando comparado os resultados apresentados pelos algoritmos Epidêmico e *PROpHET*. Enquanto o desempenho para esta métrica no trace de *Cambridge* para o algoritmo Epidêmico esteve no intervalo de 60000s a 90000s, para o algoritmo *PROpHET* o atraso médio esteve no intervalo de 1000s a 15000s. A justificativa para tal discrepância é a poda que o algoritmo *PROpHET* adota para aumentar as chances de uma mensagem ser entregue, fato que diminui a sobrecarga da rede e a quantidade de descartes, fatos que ajudam a aumentar a probabilidade de uma mensagem ser entregue em um tempo menor.

A Figura 5.13 apresenta os resultados de atraso médio de entrega no cenário *Reality*. Assim como no cenário de *Cambridge*, neste cenário a política DLK obteve novamente o melhor resultado, seguido pela política LRR. Outro fato interessante foi a diferença do desempenho destas duas políticas com relação às demais política testadas neste cenário. É interessante notar que, novamente a política MOFO obteve o pior resultado com relação ao atraso médio de entrega. Pode-se destacar também a discrepância entre os intervalos de tempo do atraso médio de entrega da combinação do algoritmo PROPhET e as políticas testadas, com relação a mesma combinação, mas com o algoritmo Epidêmico. No experimento com o algoritmo Epidêmico, no cenário

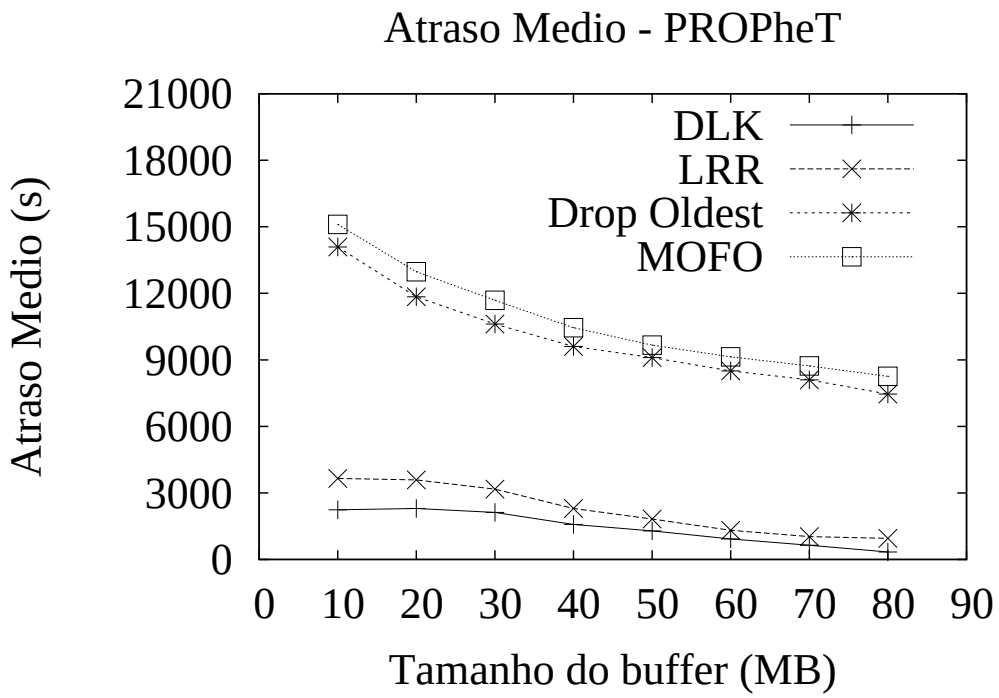

Figura 5.14. Atraso Médio utilizando roteamento *PROpHET* no trace *Reality*

Reality, o atraso médio das políticas se manteve entre 18000s e 32000s, enquanto que com o algoritmo *PROpHET* esse intervalo foi de 500s a 15000s.

É interessante notar que, em ambos os traces, a política DLK obtém o menor resultado de atraso médio de entrega. Como citado anteriormente, neste trabalho um laço forte entre dois nós é assim classificado levando em consideração, como um dos critérios, a frequência de encontros alta entre eles. Dessa maneira, mantendo-se no *buffer* dos nós mensagens destinadas a nós pertencentes a este tipo de laço, caso a frequência de encontros seja alta, as mensagens são entregues de forma mais rápida. Assim, pode-se afirmar que, a escolha de descartar do *buffer* mensagens baseando-se na relação social, pode aumentar o atraso médio nos casos em que a frequência com que se encontra nós com laços sociais fortes seja menor. Pode-se concluir, então que, para a política DLK obtenha os mesmos resultados apresentados neste trabalho com relação ao atraso médio de entrega, é ideal que seja levado em consideração na atribuição das relações sociais entre os nós da rede a frequência com que estes se encontram.

5.2.3 Média de Saltos

O número médio de saltos representa a quantidade média de vezes que uma mensagem foi repassada a outros nós até alcançar o destino final. Dessa forma, o ideal é que

a quantidade média de saltos seja a menor possível. Da mesma maneira que nos experimentos com o algoritmo Epidêmico, foram feitas duas comparações na avaliação de desempenho realizada com o algoritmo PROPhET com relação a métrica média de saltos. Primeiramente, foram comparados os resultados obtidos pela combinação do algoritmo PROPhET com cada uma políticas das políticas avaliadas, bem como uma comparação da quantidade média de saltos necessária para ser entregue uma mensagem de acordo com o seu tipo de laço social. As Figuras 5.15 e 5.16 apresentam os resultados da média de saltos nos cenários *Cambridge* e *Reality*.

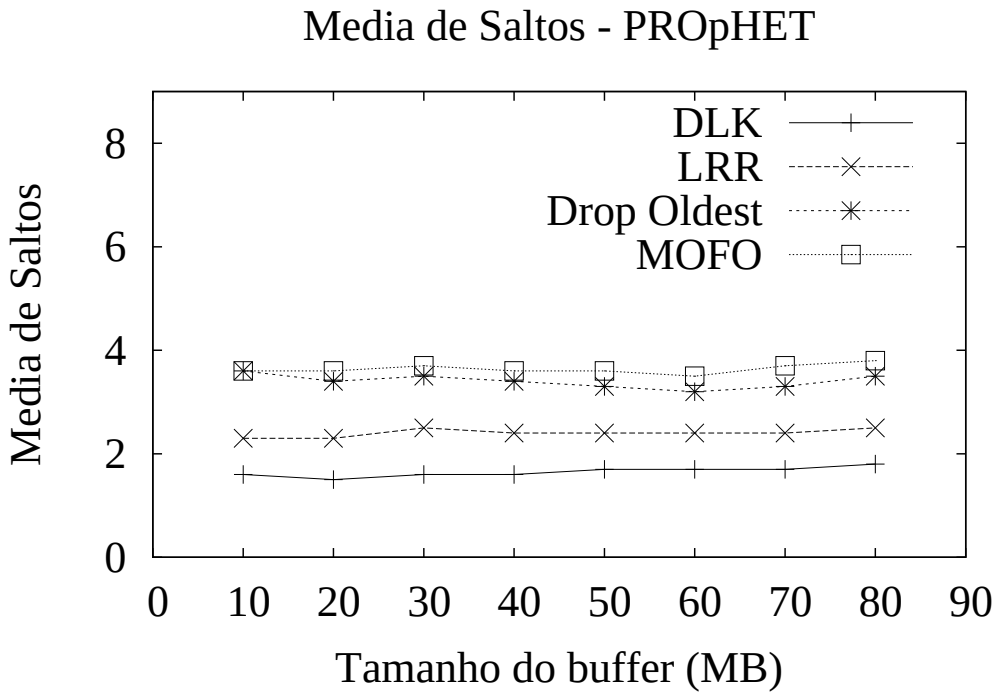

Figura 5.15. Média de Saltos utilizando roteamento *PROpHET* no trace de *Cambridge*

A Figura 5.15 apresenta os resultados da média de saltos para o cenário *Cambridge*. Neste cenário a política DLK obteve a menor média de saltos. O desempenho apresentado pela política DLK se mantém praticamente constante com relação a esta métrica. Este fato pode ser justificado pelo que citou-se acima com relação a característica da política DLK em manter no *buffer* mensagens destinadas a nós com quem se tem um laço social forte, fato que favore o encaminhamento de tais mensagens.

É interessante notar que as demais políticas também obtiveram um desempenho aparentemente linear com relação à média de saltos no cenário *Cambridge*. A justificativa para este fato está no funcionamento do algoritmo PROPhET, que realiza

replicações baseando-se em probabilidade de entrega, fato que evita que uma mensagem precise de muitos saltos para alcançar o destino.

A política MOFO obteve o pior desempenho com relação a média de saltos, mas teve um desempenho muito parecido com o da política DO. A política LRR manteve o segundo melhor desempenho, perdendo apenas para a política DLK. Pode-se destacar também o intervalo entre a quantidade média de saltos do algoritmo *PROpHET* no cenário *Cambridge*, comparado intervalo no mesmo cenário mas com o algoritmo Epidêmico. Enquanto que neste último o desempenho das políticas com relação a esta métrica esteve entre o intervalo de 2 a 14, para o algoritmo *PROpHET* este intervalo foi de 2 a 5. Mais uma vez justifica-se este fato pela poda que o algoritmo *PROpHET* realiza no encaminhamento de mensagens.

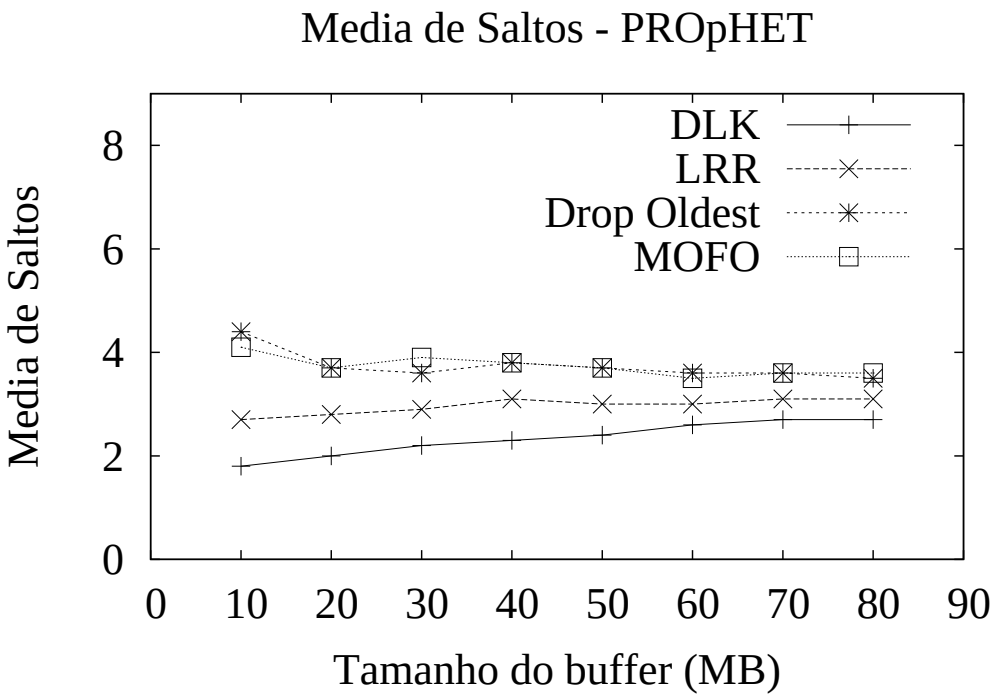

Figura 5.16. Média de Saltos utilizando roteamento *PROpHET* no trace *Reality*

No cenário *Reality* novamente a política DLK obteve o melhor desempenho obtendo uma média de saltos próxima a 2 saltos em todos os tamanhos de *buffer*. A política LRR também manteve o mesmo desempenho alcançado no trace de *Cambridge* e obteve o segundo melhor resultado. O desempenho das políticas DO e MOFO também foram parecidos novamente e em alguns pontos também foram iguais.

As Tabelas 5.3 e 5.4 indicam as média da quantidade de saltos utilizadas para entregar-se mensagens em cada tipo de força de relação social utilizada neste trabalho

Tabela 5.3. Média de saltos apresentado pela política DLK no trace *Reality*

Tipo	10 MB	20 MB	30 MB	40 MB	50 MB	60 MB	70 MB	80 MB
Laço Forte	1,7	1,9	2	2,1	2,1	2,3	2,4	2,5
Laço Médio	2	2,3	2,6	2,5	2,7	2,7	2,8	2,9
Laço Fraco	2,8	2,5	3	3,1	3,4	3,4	3,6	3,4

Tabela 5.4. Média de saltos apresentado pela política DLK no trace *Cambridge*

Tipo	10 MB	20 MB	30 MB	40 MB	50 MB	60 MB	70 MB	80 MB
Laço Forte	1,5	1,4	1,5	1,5	1,5	1,6	1,5	1,6
Laço Médio	1,6	1,7	1,8	1,8	2	1,8	1,9	1,9
Laço Fraco	2	2	2,1	2,3	2,5	2,7	2,9	3

de forma individual, em cada cenário. Os dados apresentados nesta Tabelas exemplificam o fato citado anteriormente de que a estratégia de descarte da política DLK favorece a diminuição da média de saltos. Pode-se ainda perceber, através destes resultados, a influência da força do laço social na quantidade de saltos utilizada. Para o tipo de laço social Forte, essa média é a menor obtida variando entre 1,7 saltos e 2,5 saltos, valores relativamente baixos. O tipo de laço Médio, por sua vez, obteve um resultado inferior ao do tipo de laço mais Forte, tendo a média de saltos variando entre 2,8 saltos e 3,4 saltos. Essas diferenças indicam que quanto maior a força do laço social, menor será a quantidade de saltos utilizadas pois mensagens destinadas a estes nós tendem a serem mantidas no *buffer*, favorecendo assim o seu envio ao destino final.

A característica do descarte feito pela política DLK tende a minimizar a quantidade média de saltos necessária para que uma mensagem chegue ao destino final. A medida que se descarta mensagens destinadas a usuários menos conhecidos , se mantém no *buffer* mensagens destinadas a usuários com quem se mantém uma relação social mais forte, e a mensagem é entregue ao destino utilizando-se menos encaminhamentos.

É interessante destacar que, o desempenho da política DLK com relação a métrica média de saltos, diferentemente da métrica atraso médio, não depende diretamente da quantidade de contatos entre os nós. A ideia é que DLK mantenha a média de saltos relativamente baixa, mais ainda quando combinada com o algoritmo *PROpHET*, o qual realiza o encaminhamento de forma probabilística, fato que diminui a quantidade de mensagens enviadas na rede e assim diminui a sobrecarga da mesma.

Capítulo 6

Conclusões

O grande desafio em DTNs, está em fornecer um serviço de rede de comunicação de dados efiente, mesmo com os longos atrasos e as frequentes desconexões. Neste sentido, percebe-se que as pesquisas neste tipo de rede tem maior foco no desenvolvimento de novos algoritmos de roteamento ou refinamentos dos já existentes, com objetivo de aumentar a eficiência na entrega de mensagens. É bastante perceptível na literatura para DTNs um esforço menor das pesquisas com relação a outros tópicos, como o gerenciamento do *buffer* e o gerenciamento de energia, fatores que influenciam diretamente o roteamento e, consequentemente, afetam a eficiência da rede.

Ainda que seja visto uma quantidade menor de trabalhos, existem diversos trabalhos com propostas de algoritmos dedicados a gerência de *buffer* para redes DTN. De maneira geral, estes algoritmos gerenciam o *buffer* utilizando informações parciais do estado da rede ou informações das mensagens que estão no *buffer*. No entanto, a escolha feita por essas políticas, tomando como base apenas informações locais ou globais da rede, pode ser prejudicial, visto que pode selecionar para o descarte mensagens que o usuário gostaria de encaminhar ou carregar consigo. Esta vontade do usuário pode ser representada pelo altruísmo que o mesmo tem para com os outros usuários da rede, fator que influencia diretamente na sua vontade em carregar ou não mensagens para outro usuário.

Neste trabalho, apresentou-se uma política de gerência de *buffer* para redes DTN baseada em relações sociais. Partindo-se da suposição de que no mundo real os usuários são socialmente egoístas e desejam ajudar usuários de acordo com a força da relação social entre eles, o objetivo principal da política, é indicar, no caso de *overflow*, mensagens destinadas a outros usuários com quem se tem uma relação social mais fraca.

Pode-se destacar duas contribuições principais com a criação da política DLK: primeiramente, atender ao perfil egoísta dos usuários reais, fato que representa uma

escolha mais correta por respeitar a vontade do usuário sobre que mensagem descartar em casos de *overflow*. Segundo, a política DLK utiliza como base para o descarte de uma mensagem características sociais, fator este que tem sido bastante discutido com relação ao roteamento de mensagens, mas, com relação a gerência de *buffer* para DTNs ainda não havia sido abordado. Com este trabalho pretende-se chamar a atenção da comunidade científica para discutir-se o uso de características sociais também na gerência do *buffer* para DTNs.

Para avaliar-se a política realizou-se experimentos visando avaliar métricas de entrega de mensagens em dois cenários baseados em movimentação humana. Os resultados obtidos mostraram que em ambos os cenários avaliados o desempenho foi muito promissor superando as outras políticas de gerência de *buffer* testadas.

A política proposta obteve uma maior taxa de entrega, alcançando até 15% mais mensagens entregues que a política de segundo melhor desempenho, com um menor atraso médio de entrega e com uma menor quantidade de saltos.

Em trabalhos futuros, pretende-se investigar-se o hibridismo entre métodos existentes na literatura e o método DLK, investigar-se as influências dos variados cenários de redes DTN no desempenho de DLK e dos principais algoritmos para gerência de *buffer* existentes, além de buscar-se alternativas para otimizar o método DLK.

Referências Bibliográficas

(2013). Projeto internet interplanet?ria.

Ayub, Q. & Rashid, S. (2010). T-drop: An optimal buffer management policy to improve qos in dtn routing protocols. *Journal of Computing*, 2(10):46--50.

Azevedo, T. S. (2010). *AJUSTE ADAPTATIVO DO SINAL DE TRANSMISSÃO DE MENSAGENS DE DESCOBERTA PARA CONSUMO EFICIENTE DE ENERGIA EM REDES TOLERANTES A ATRASOS E INTERRUPÇÕES.* Tese de doutorado, Universidade Federal do Rio de Janeiro.

Burgess, J.; Gallagher, B.; Jensen, D. & Levine, B. N. (2006). Maxprop: Routing for vehicle-based disruption-tolerant networks. pp. 1 –11,.

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

Chen, P.-A. & Kempe, D. (2008). Altruism, selfishness, and spite in traffic routing. Em *Proceedings of the 9th ACM conference on Electronic commerce*, pp. 140--149. ACM.

de Oliveira, C. T.; Moreira, M. D.; Rubinstein, M. G.; Costa, L. H. M. & Duarte, O. C. M. (2007). Redes tolerantes a atrasos e desconexoes. *SBRC Simpósio Brasieliro de Redes de Computadores e Sistemas Distribuıdos*.

Diot, C. et al. (2004). Haggle project.

Eagle, N. & 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. Em *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 27--34. ACM.

Farrell, S.; Cahill, V.; Geraghty, D.; Humphreys, I. & McDonald, P. (2006). When tcp breaks: Delay-and disruption-tolerant networking. *Internet Computing, IEEE*, 10(4):72--78.

Fathima, G. & Wahidabanu, R. (2008). Buffer management for preferential delivery in opportunistic delay tolerant networks. *International Journal of Wireless & Mobile Networks (IJWMN)*, 3(5):15--28.

Franklin, W. W. & White, K. (2008). Stationarity tests and mser-5: exploring the intuition behind mean-squared-error-reduction in detecting and correcting initialization bias. Em *Simulation Conference, 2008. WSC 2008. Winter*, pp. 541--546. IEEE.

Hyytiä, E.; Koskinen, H.; Lassila, P.; Penttinen, A.; Roszik, J. & Virtamo, J. (2005). Random waypoint model in wireless networks. *Networks and Algorithms: complexity in Physics and Computer Science, Helsinki*.

Juang, P.; Oki, H.; Wang, Y.; Martonosi, M.; Peh, L. S. & Rubenstein, D. (2002). Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. *SIGOPS Oper. Syst. Rev.*, 36(5):96--107.

Jun, H. (2007). Power management in disruption tolerant networks.

Jun, H.; Ammar, M. H.; Corner, M. D. & Zegura, E. W. (2006). Hierarchical power management in disruption tolerant networks with traffic-aware optimization. Em *Proceedings of the 2006 SIGCOMM workshop on Challenged networks*, pp. 245--252. ACM.

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

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

Li, Q.; Zhu, S. & Cao, G. (2010). Routing in socially selfish delay tolerant networks. Em *INFOCOM, 2010 Proceedings IEEE*, pp. 1--9. IEEE.

Li, Y.; Zhao, L.; Liu, Z. & Liu, Q. (2009). N-drop: congestion control strategy under epidemic routing in dtn. Em *Proceedings of the 2009 International Conference on*

Wireless Communications and Mobile Computing: Connecting the World Wirelessly, pp. 457--460. ACM.

Lindgren, A.; Doria, A. & Schelén, O. (2003). Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(3):19--20.

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

Liu, Y.; Wang, J.; Zhang, S. & Zhou, H. (2011). A buffer management scheme based on message transmission status in delay tolerant networks. Em *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pp. 1 –5.

Meketon, M. S. & Schmeiser, B. (1984). Overlapping batch means: Something for nothing? Em *Proceedings of the 16th conference on Winter simulation*, pp. 226--230. IEEE Press.

Mota, E. d. S. (2002). *Performance of Sequential Batching-based Methods of Output Data Analysis in Distributed Steady-state Stochastic Simulation*. Tese de doutorado, Universitätsbibliothek.

Naves, Juliano F., I. M. M. C. V. N. d. A. (2012). Lps e lrf: Politicas de gerenciamento de buffer eficientes para redes tolerantes a atrasos e desconexoes. *Simpósio Brasileiro de Redes de Computadores (SBRC?12)*, pp. 293--305.

Neto, J. B. P. (2012). *Um Modelo para Previs ?o do Volume de Contato em Redes Tolerantes a Atrasos e Desconex?es: Uma Abordagem Quantitativa*. Tese de doutorado, UNIVERSIDADE FEDERAL DO AMAZONAS.

Okasha, S. (2005). Altruism, group selection and correlated interaction. *The British journal for the philosophy of science*, 56(4):703--725.

Oliveira, C. T. & Duarte, O. (2007). Uma análise da probabilidade de entrega de mensagens em redes tolerantes a atrasos e desconexoes. *Simpósio Brasileiro de Redes de Computadores (SBRC?07)*, pp. 293--305.

Oliveira, C. T. d. (2008). *Uma Proposta de Roteamento Probabilístico para Redes Tolerantes a Atrasos e Desconexões*. Tese de doutorado, UNIVERSIDADE FEDERAL DO RIO DE JANEIRO.

Oliveira, E. C. & de Albuquerque, C. V. (2009). Nectar: a dtn routing protocol based on neighborhood contact history. Em *Proceedings of the 2009 ACM symposium on Applied Computing*, pp. 40--46. ACM.

Postel, J. (2003). Rfc 793: Transmission control protocol, september 1981. *Status: Standard*.

Rashid, S. & Ayub, Q. (2010). Effective buffer management policy dla for dtn routing protocals under congetion. *International Journal of Computer and Network Security*, 2(9):118--121.

Scott, K. L. & Burleigh, S. (2007). Bundle protocol specification.

Soares, V.; Rodrigues, J.; Ferreira, P. & Nogueira, A. (2009). Improvement of messages delivery time on vehicular delay-tolerant networks. Em *Parallel Processing Workshops, 2009. ICPPW '09. International Conference on*, pp. 344 –349.

Spyropoulos, T.; Psounis, K. & Raghavendra, C. S. (2005). Spray and wait: an efficient routing scheme for intermittently connected mobile networks. Em *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pp. 252--259. ACM.

TANG, L.; CHAI, Y.; LI, Y. & 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. Relatório técnico, Technical Report CS-200006, Duke University.

Warthman, F. (2007). Delay-tolerant networks (dtns): A tutorial v1. 1, march, 2003.

Xu, K.; Hui, P.; K.Li, V. O.; Crowcroft, J.; Latora, V. & Lio, P. (2009). Impact of altruism on opportunistic communications. Em *Proceedings of the first international conference on Ubiquitous and future networks*, ICUFN'09, pp. 153--158, Piscataway, NJ, USA.

Zhang, X.; Neglia, G.; Kurose, J. & Towsley, D. (2007). Performance modeling of epidemic routing. *Computer Networks*, 51(10):2867--2891.

Zhu, Y.; Xu, B.; Shi, X. & Wang, Y. (2012). A survey of social-based routing in delay tolerant networks: positive and negative social effects.

Apêndice A

Códigos auxiliares implementados na pesquisa

O código a seguir foi escrito na linguagem JAVA como um método da classe Active-Router do simulador the One. Maiores detalhes sobre o funcionamento da política são dados no capítulo 3 do presente trabalho.

A.1 Métodos estatísticos

A.1.1 Implementação do método MSER-5

```
/*
    * ===== MSER-5 =====
    *
    * - n: número de observações;
    * - b: número de batches;
    * - m: tamanho dos batches.
    *
    * Obs.: Considerando n um múltiplo de 5 e m por padrão igual a 5.
    * Obs.: O número de batches aumenta conforme o aumento do número de amostras.
    */
    public class MSER
    /*
    * Take the samples and divide them in batches of length 5. The number os batches is b = n/m.
```

```
*/
private static ArrayList<Double> makeBatches (int n, int m)
ArrayList<Double> batches = new ArrayList<Double>();
ArrayList<Double> samples;
int i, j, b;
double avrg = 0.0;
b = n/m;
samples = Statistic.getSamples();
for (i = 0; i < b; i++)
avrg = 0.0;
for (j = i*m; j < i*m + m; j++)
avrg += samples.get(j);
batches.add(avrg/m);
return batches;
/*
* Return the average of batches. They are idexed by zero.
*
* Obs.: If the number of batches is very low, and the d value is equal o
* greater than that, the division will by zero or a negative number!
*/
private static double batchAverage (ArrayList<Double> batches, int d)
double avrg = 0.0;
int i;
for (i = d+1; i < batches.size(); i++)
avrg += batches.get(i);
return avrg / (batches.size()-d-1);
/*
* Use MSER-5 to determine the transient.
*
* First, create the batches, then, compute their average. After that,
* compute the minimum d' for the data.
*
* It was assume that d' can be in the first, the second, until the last 5
* batches (i.e.: 1, 2, 3, ..., 15 (last iteration), ..., 20). If the value
* found is in the right side of the half of batches used, so we need to get
* more 10% of data, otherwise, minimum d' is in the left side, we can stop
* the MSER-5 and "remove"all the data before this d'.
```

```
*/
public static int mser5()
ArrayList<Double> batches;
int i, j, n, m, d, dmin;
double avrg = 0.0, batchavrg, sum, tmp, dminvalue = -1;
/* Default Values */
m = 5;
dmin = 0;
n = Statistic.getN(); /* Make n a multiple of 5 */
while (n%5 != 0)
n–;
batches = makeBatches(n, m);
/*
* To stop MSER-5 it's necessary to reach the last 5 batches, which
* means stop before the last 25 data collected.
*/
dmin = n/2;
for (d = 0; d < batches.size() - m; d++)
batchavrg = batchAverage (batches, d);
sum = 0.0;
/* the batches are indexed by 0 */
for (j = d+1; j < batches.size(); j++)
tmp = batches.get(j) - batchavrg;
tmp = Math.pow(tmp, 2);
sum += tmp;
if (sum == 0) continue;
n = batches.size();
tmp = Math.pow(n - d-1, 2);
tmp = (1/tmp) * sum;
if (d == 0 || dminvalue == -1)
dmin = d+1;
dminvalue = tmp;
else if (tmp <= dminvalue)
dmin = d+1;
dminvalue = tmp;
if (dmin >= n/2)
n = Statistic.getN();
```

```
n += n*(0.1);
Statistic.setN(n);
return -1;
else
n = Statistic.getN();
Statistic.setN(n+dmin*5); /*coletar mais dmin*5 dados referentes
*aos elementos logicamente removidos.
*/
return dmin*5; /* because each batch stores 5 datas */
```

A.1.2 Implementação do método OBM

```
public class OBM
    private static int B = 10, M = 0;
    private static int OVERLAPDEGREE;
    private static double GAMA = 0.05, H;
    private static double t;
    public static double getH()  return H;
    /* Take the samples and divide them in N-M+1 batches of length M. */ private
static ArrayList<Double> makeBatches (int n, int dmin)
    ArrayList<Double> batches = new ArrayList<Double>();
    ArrayList<Double> samples;
    int i, j = 0, m, step;
    double avrg = 0.0;
    samples = Statistic.getSamples();
    m = M;
    step = OVERLAPDEGREE;
    n = n - dmin;
    /* se o número de elementos coletados ainda formar uma amostra
    * i -> índice da amostra e o número de batches formados;
    * step -> grau de superposição;
    * m -> comprimento dos batches.
    */
    for (i = 0; i+m < n; i += step)
    avrg = 0.0;
    for (j = dmin + i; j < dmin + i + m; j++)
    avrg += samples.get(j);
```

```
batches.add(avrg/m);
return batches;
/* calcula a média total dos blocos */
private static double batchesaverage (ArrayList<Double> batches)
int b;
double avrg = 0;
for (Double batch: batches)
avrg += batch;
b = batches.size();
return avrg/b;
private static double computevariance (double batchesavrg, ArrayList<Double>
batches)
int i, b;
double sum = 0;
b = batches.size();
for (i = 0; i < b; i++)
sum += Math.pow(batches.get(i) - batchesavrg, 2);
sum = sum / (b*(b-1));
return Math.sqrt(sum); /*standard deviation*/
/* * função principal para o método OBM * n é sempre o tamanho total da
amostra incluindo o transiente. */
public static boolean computeOBM (int n, int dmin)
double batchesavrg, variance;
ArrayList<Double> batches;
if (M == 0)
M = Correlacao.getM();
OVERLAPDEGREE = 1;
batches = makeBatches(n, dmin);
batchesavrg = batchesaverage(batches);
variance = computevariance(batchesavrg, batches);
t = TDistribution.tdistribution(n-dmin-1, 0.025);
H = t*variance; /* largura do intervalo de confiança */
if (batchesavrg != 0 H/batchesavrg <= GAMA)
return true;
else
//M += Statistic.getM(); /* aumenta M */
M += 5; /* aumenta M */
```

```
n = B*M; /* amostras necessárias */
Statistic.setN(n); /* seta o novo valor */
return false;
```

A.1.3 Implementação da classe que elimina a correlação

```
public class Correlacao
    /* Default Values */
    private static final double BETA = 0.62; // para testar o algoritmo, usar 1.3
    public static double rnvmin = -1;
    private static int B = 10, M = 0;
    public static int getM()  return M;
    /* Take the samples and divide them in 10 batches of length M. */
    private static ArrayList<Double> makeBatches (int dmin)
    ArrayList<Double> batches = new ArrayList<Double>();
    ArrayList<Double> samples;
    int i, j = 0, m;
    double avrg = 0.0;
    samples = Statistic.getSamples();
    m = M;
    for (i = 0; i < B; i++)
    avrg = 0.0;
    for (j = i*m + dmin; j < i*m + dmin + m; j++)
    avrg += samples.get(j);
    batches.add(avrg/m);
    return batches;
    /* Compute von Neuman */ public static boolean compute(int n, int dmin)
    int i, b;
    double batchesavrg, rnv, rnvDen;
    ArrayList<Double> batches;
    if (M == 0) M = Statistic.getM();
    b = B;
    while (n-dmin >= B*M)  /* in the first time, n can be greater than B*M */
    batches = makeBatches(dmin);
    /* Média total dos batches */
    batchesavrg = 0;
    for(i = 0; i < b; i++)
```

```
batchesavrg += batches.get(i);
batchesavrg /= b;
/* calcular o resultado dos somatórios da divisão */
rnv = rnvDen = 0;
for(i = 0; i < b-1; i++)
rnv += Math.pow(batches.get(i) - batches.get(i+1), 2);
rnvDen += Math.pow(batches.get(i) - batchesavrg, 2);
/* Última iteração da parte inferior */
rnvDen += Math.pow(batches.get(i) - batchesavrg, 2);
if (rnvDen == 0)
M += Statistic.getM();
continue;
rnv = rnv / rnvDen;
if (rnvmin == -1 || rnv < rnvmin)
rnvmin = rnv;
if (rnv > BETA) /* não satisfez */
M += Statistic.getM();
else
return false;
n = B*M; /*novo valor de amostras necessárias*/
Statistic.setN(n); /* seta o novo valor */
return true;
```

A.2 Arquivos de configuração utilizados

A.2.1 Arquivo de configuração usado para o trace de *Cambridge* com o roteamento Epidêmico

```
#
# Arquivo de Configuração The One Simulator #
Scenario.name = CambridgeEpidemico
Scenario.simulateConnections = false
Scenario.updateInterval = 1
Scenario.endTime = 987529
type : which interface class the interface belongs to
For different types, the sub-parameters are interface-specific
```

```
For SimpleBroadcastInterface, the parameters are:
transmitSpeed : transmit speed of the interface (bytes per second)
transmitRange : range of the interface (meters)*/
btInterface.type = SimpleBroadcastInterface
btInterface.transmitSpeed = 250k
btInterface.transmitRange = 10
highspeedInterface.type = SimpleBroadcastInterface
highspeedInterface.transmitSpeed = 10M
highspeedInterface.transmitRange = 1000
Scenario.nrofHostGroups = 1
Group-specific settings:
Group.movementModel = RandomWaypoint
Group.router = ProphetRouter
Group.bufferSize = 80M
Group.nrofInterfaces = 1
Group.interface1 = btInterface
Group.speed = 0.5, 1.5
Group.nrofHosts = 36
Group1.groupID = p
Events.nrof = 2
Events1.class = MessageEventGenerator
Events1.interval = 180,180
Events1.size = 100k, 100k
Events1.hosts = 0,35
Events1.prefix = M
Events2.filePath = mobilidade/cambridgemodel.txt
MovementModel.rngSeed = 971
MovementModel.warmup = 0
MapBasedMovement.nrofMapFiles = 4
MapBasedMovement.mapFile1 = data/roads.wkt
MapBasedMovement.mapFile2 = data/mainroads.wkt
MapBasedMovement.mapFile3 = data/pedestrianpaths.wkt
MapBasedMovement.mapFile4 = data/shops.wkt
Report.nrofReports = 8
Report.warmup = 0
Report.reportDir = reports/
Report.report1 = DeliveredMessagesReport
```

Report.report2 = CreatedMessagesReport
Report.report3 = MessageStatsReport
Report.report4 = EventLogReport
Report.report5 = MessageDelayReport
Report.report6 = MessageDeliveryReport
Report.report7 = MessageReport
Report.report8 = MessageGraphvizReport
ProphetRouter.secondsInTimeUnit = 30
SprayAndWaitRouter.nrofCopies = 6
SprayAndWaitRouter.binaryMode = true
Optimization.cellSizeMult = 5
Optimization.randomizeUpdateOrder = true
GUI.UnderlayImage.fileName = data/helsinkiunderlay.png
GUI.UnderlayImage.offset = 64, 20
GUI.UnderlayImage.scale = 4.75
GUI.UnderlayImage.rotate = -0.015
GUI.EventLogPanel.nrofEvents = 100

A.2.2 Arquivo de configuração usado para o trace de *Cambridge* com o roteamento PROPhET

# Arquivo de Configuração The One Simulator #
Scenario.name = *Cambridge*PROPhET
Scenario.simulateConnections = false
Scenario.updateInterval = 1
Scenario.endTime = 987529
type : which interface class the interface belongs to
For different types, the sub-parameters are interface-specific
For SimpleBroadcastInterface, the parameters are:
transmitSpeed : transmit speed of the interface (bytes per second)
transmitRange : range of the interface (meters)*/
btInterface.type = SimpleBroadcastInterface
btInterface.transmitSpeed = 250k
btInterface.transmitRange = 10
highspeedInterface.type = SimpleBroadcastInterface
highspeedInterface.transmitSpeed = 10M
highspeedInterface.transmitRange = 1000

```
Scenario.nrofHostGroups = 1
Group-specific settings:
Group.movementModel = RandomWaypoint
Group.router = ProphetRouter
Group.bufferSize = 80M
Group.nrofInterfaces = 1
Group.interface1 = btInterface
Group.speed = 0.5, 1.5
Group.nrofHosts = 41
Group1.groupID = p
Events.nrof = 2
Events1.class = MessageEventGenerator
Events1.interval = 180,180
Events1.size = 100k, 100k
Events1.hosts = 0,41
Events1.prefix = M
Events2.filePath = mobilidade/cambridgemodel.txt
MovementModel.rngSeed = 971
MovementModel.warmup = 0
MapBasedMovement.nrofMapFiles = 4
MapBasedMovement.mapFile1 = data/roads.wkt
MapBasedMovement.mapFile2 = data/mainroads.wkt
MapBasedMovement.mapFile3 = data/pedestrianpaths.wkt
MapBasedMovement.mapFile4 = data/shops.wkt
Report.nrofReports = 8
Report.warmup = 0
Report.reportDir = reports/
Report.report1 = DeliveredMessagesReport
Report.report2 = CreatedMessagesReport
Report.report3 = MessageStatsReport
Report.report4 = EventLogReport
Report.report5 = MessageDelayReport
Report.report6 = MessageDeliveryReport
Report.report7 = MessageReport
Report.report8 = MessageGraphvizReport
ProphetRouter.secondsInTimeUnit = 30
SprayAndWaitRouter.nrofCopies = 6
```

SprayAndWaitRouter.binaryMode = true
Optimization.cellSizeMult = 5
Optimization.randomizeUpdateOrder = true
GUI.UnderlayImage.fileName = data/helsinkiunderlay.png
GUI.UnderlayImage.offset = 64, 20
GUI.UnderlayImage.scale = 4.75
GUI.UnderlayImage.rotate = -0.015
GUI.EventLogPanel.nrofEvents = 100

A.2.3 Arquivo de configuração usado para o trace de reality com o roteamento Epidêmico

#
# Arquivo de Configuração The One Simulator #
Scenario.name = realityEpidemico
Scenario.simulateConnections = false
Scenario.updateInterval = 1
Scenario.endTime = 987529
type : which interface class the interface belongs to
For different types, the sub-parameters are interface-specific
For SimpleBroadcastInterface, the parameters are:
transmitSpeed : transmit speed of the interface (bytes per second)
transmitRange : range of the interface (meters)*/
btInterface.type = SimpleBroadcastInterface
btInterface.transmitSpeed = 250k
btInterface.transmitRange = 10
highspeedInterface.type = SimpleBroadcastInterface
highspeedInterface.transmitSpeed = 10M
highspeedInterface.transmitRange = 1000
Scenario.nrofHostGroups = 1
Group-specific settings:
Group.movementModel = RandomWaypoint
Group.router = ProphetRouter
Group.bufferSize = 80M
Group.nrofInterfaces = 1
Group.interface1 = btInterface
Group.speed = 0.5, 1.5

```
Group.nrofHosts = 36
Group1.groupID = p
Events.nrof = 2
Events1.class = MessageEventGenerator
Events1.interval = 180,180
Events1.size = 100k, 100k
Events1.hosts = 0,35
Events1.prefix = M
Events2.filePath = mobilidade/cambridgemodel.txt
MovementModel.rngSeed = 971
MovementModel.warmup = 0
MapBasedMovement.nrofMapFiles = 4
MapBasedMovement.mapFile1 = data/roads.wkt
MapBasedMovement.mapFile2 = data/mainroads.wkt
MapBasedMovement.mapFile3 = data/pedestrianpaths.wkt
MapBasedMovement.mapFile4 = data/shops.wkt
Report.nrofReports = 8
Report.warmup = 0
Report.reportDir = reports/
Report.report1 = DeliveredMessagesReport
Report.report2 = CreatedMessagesReport
Report.report3 = MessageStatsReport
Report.report4 = EventLogReport
Report.report5 = MessageDelayReport
Report.report6 = MessageDeliveryReport
Report.report7 = MessageReport
Report.report8 = MessageGraphvizReport
ProphetRouter.secondsInTimeUnit = 30
SprayAndWaitRouter.nrofCopies = 6
SprayAndWaitRouter.binaryMode = true
Optimization.cellSizeMult = 5
Optimization.randomizeUpdateOrder = true
GUI.UnderlayImage.fileName = data/helsinkiunderlay.png
GUI.UnderlayImage.offset = 64, 20
GUI.UnderlayImage.scale = 4.75
GUI.UnderlayImage.rotate = -0.015
GUI.EventLogPanel.nrofEvents = 100
```

A.2.4 Arquivo de configuração usado para o trace de reality com o roteamento PROPhET

```
# Arquivo de Configuração The One Simulator #
    Scenario.name = realityPROPhET
    Scenario.simulateConnections = false
    Scenario.updateInterval = 1
    Scenario.endTime = 987529
    type : which interface class the interface belongs to
    For different types, the sub-parameters are interface-specific
    For SimpleBroadcastInterface, the parameters are:
    transmitSpeed : transmit speed of the interface (bytes per second)
    transmitRange : range of the interface (meters)*/
    btInterface.type = SimpleBroadcastInterface
    btInterface.transmitSpeed = 250k
    btInterface.transmitRange = 10
    highspeedInterface.type = SimpleBroadcastInterface
    highspeedInterface.transmitSpeed = 10M
    highspeedInterface.transmitRange = 1000
    Scenario.nrofHostGroups = 1
    Group-specific settings:
    Group.movementModel = RandomWaypoint
    Group.router = ProphetRouter
    Group.bufferSize = 80M
    Group.nrofInterfaces = 1
    Group.interface1 = btInterface
    Group.speed = 0.5, 1.5
    Group.nrofHosts = 41
    Group1.groupID = p
    Events.nrof = 2
    Events1.class = MessageEventGenerator
    Events1.interval = 180,180
    Events1.size = 100k, 100k
    Events1.hosts = 0,41
    Events1.prefix = M
    Events2.filePath = mobilidade/cambridgemodel.txt
    MovementModel.rngSeed = 971
```

```
MovementModel.warmup = 0
métricas MapBasedMovement.nrofMapFiles = 4
MapBasedMovement.mapFile1 = data/roads.wkt
MapBasedMovement.mapFile2 = data/mainroads.wkt
MapBasedMovement.mapFile3 = data/pedestrianpaths.wkt
MapBasedMovement.mapFile4 = data/shops.wkt
Report.nrofReports = 8
Report.warmup = 0
Report.reportDir = reports/
Report.report1 = DeliveredMessagesReport
Report.report2 = CreatedMessagesReport
Report.report3 = MessageStatsReport
Report.report4 = EventLogReport
Report.report5 = MessageDelayReport
Report.report6 = MessageDeliveryReport
Report.report7 = MessageReport
Report.report8 = MessageGraphvizReport
ProphetRouter.secondsInTimeUnit = 30
SprayAndWaitRouter.nrofCopies = 6
SprayAndWaitRouter.binaryMode = true
Optimization.cellSizeMult = 5
Optimization.randomizeUpdateOrder = true
GUI.UnderlayImage.fileName = data/helsinkiunderlay.png
GUI.UnderlayImage.offset = 64, 20
GUI.UnderlayImage.scale = 4.75
GUI.UnderlayImage.rotate = -0.015
GUI.EventLogPanel.nrofEvents = 100
```