DISSEMINAÇÃO DE MENSAGENS EM REDES OPORTUNISTAS BASEADA EM RELAÇÕES SOCIAIS E APRENDIZAGEM DE MÁQUINA

CAMILO BATISTA DE SOUZA

DISSEMINAÇÃO DE MENSAGENS EM REDES OPORTUNISTAS BASEADA EM RELAÇÕES SOCIAIS E APRENDIZAGEM DE MÁQUINA

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal do Amazonas como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Orientador: Edjair de Souza Mota

Manaus

Agosto de 2019

Souza, Camilo Batista de
S729d Disseminação de mensagens em Redes Oportunistas baseada em relações sociais e aprendizagem de máquina / Camilo Batista de Souza. 2019
113 f.: 31 cm.

Orientador: Edjair de Souza Mota
Tese (Doutorado em Informática) - Universidade Federal do Amazonas

1. Redes Oportunistas. 2. Aprendizagem de máquina. 3. disseminação de mensagens. 4. Relacionamentos sociais. I. Mota, Edjair de Souza II. Universidade Federal do Amazonas III. Título

FOLHA DE APROVAÇÃO

"Disseminação de Mensagens em Redes Oportunistas Baseada em Relações Sociais e Aprendizagem de Máquina"

CAMILO BATISTA DE SOUZA

Tese de Doutorado defendida e aprovada pela banca examinadora constituída pelos Professores:

Prof. Edjair Souza Mota - PRESIDENTE

Prof. Leandro Silva Galvão de Carvalho - MEMBRO EXTERNO

Prof. Celso Barbosa Carvalho - MEMBRO EXTERNO

Prof. José Neuman de Souza - MEMBRO EXTERNO

Prof. Carlos Tavares Calafate - MEMBRO EXTERNO

Manaus, 30 de Agosto de 2019

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

Agradecimentos

Ao final da realização da presente tese de doutorado, agradeço primeiramente a Deus por permitir que concluísse todos os passos necessários para realização do trabalho.

Gostaria de dedicar este trabalho e agradecer especialmente a duas pessoas. Minha querida avó, Olimpia Batista de Souza. Gostaria que você estivesse aqui nesse dia para ver o seu neto nesse dia tão especial. Você esteve presente no início dessa caminhada, gostaria que você estivesse aqui neste final para dizermos juntos: Ufa, conseguimos. Mas, de certa forma você está aqui conosco. Minha querida mãe, Rosângela. Acredito que qualquer coisa que escreva aqui não seria suficiente para explicar o quanto você é importante não somente nesse trabalho, mas em toda a minha vida. Gostaria de agradecer por ter me dado a oportunidade de estudar. Você e eu sabemos o quanto foi duro chegar até aqui. Talvez eu nem saiba em sua totalidade o quanto você se doou para que eu tivesse essa oportunidade. Nunca esquecerei do dia que liguei dizendo que deixaria para estudar depois, porque precisava de dinheiro para pagar as contas e isso era mais urgente naquele momento. E você me disse que não, que eu aguentasse, que aquilo era passageiro, que nós íamos vencer. Essa foi uma das grandes lições que você me ensinou. São tantas, que me atrevo a dizer que a verdadeira doutora aqui é você. Obrigado, mãe. Te amarei enquanto viver. Agradeço à minha família, razão da minha vida. Amo vocês do fundo do meu coração.

Agradeço ao meu orientador, professor Dr. Edjair Mota. Obrigado por esses quase 10 anos de muito trabalho. Obrigado por ter me mostrado como realizar pesquisa científica com qualidade. Obrigado pelas lições de vida que me destes. Obrigado por não ter desistido de mim. Sempre me lembrarei das noites que, mesmo sem necessitar, dedicaste a passar em claro para me ajudar no início do desenvolvimento das simulações realizadas neste trabalho. Você me ensinou uma grande lição como docente: o verdadeiro papel do professor não é ensinar algo a um aluno que sabe muita coisa, mas sim ajudar aos alunos que têm dificuldade em entender certos assuntos a se desenvolverem. Sempre lembrarei dessa lição.

Agradeço aos professores da UPV/Valência, professores Pietro Manzoni, Carlos

Calafate, Juan-Carlos Cano, e Enrique-Orallo. Obrigado a todas as lições que me ensinaram. Pela paciência que tiveram nas orientações e ajustes necessários nos artigos escritos. Obrigado pela oportunidade de realizar o estágio sanduíche no grupo GRC da UPV. Agradeço também aos colegas de grupo do GRCM/UFAM e do GRC/UPV. Obrigado por todo companheirismo, pelas risadas, pelos conselhos, por tudo. Sempre lembrarei de todos os momentos vividos.

A todos os meus amigos, os "Pacatos Guerreiros", o meu muito obrigado por todo o companheirismo, amizade, pelas risadas, compreensão, e principalmente por todos os "churras"realizados nesses anos. Obrigado à Nicole Oliveira por compreender os momentos de ausência necessários para a realização dessa tese de doutorado.

A todos que direta ou indiretamente contribuíram para que esse sonho se tornasse realidade, o meu muito obrigado.

"Sonhos determinam o que você quer.
Ação determina o que você conquista."
(Aldo Novak)

Resumo

Redes Oportunistas estão se tornando uma solução para fornecer suporte de comunicação em áreas com redes celulares sobrecarregadas, e em cenários onde uma infraestrutura fixa não está disponível, como em regiões remotas e em desenvolvimento. Uma questão crítica, que ainda requer uma solução satisfatória, é o projeto de uma solução eficiente de disseminação de dados em termos da taxa de entrega, atraso médio e custo de encaminhamentos. Para solucionar esse problema, a maioria dos pesquisadores tem usado o estado da rede ou a mobilidade dos nós como um critério para a disseminação dos dados. Recentemente, soluções baseadas em relacionamentos sociais têm sido consideradas como uma alternativa promissora.

Seguindo a filosofia dessa nova categoria de protocolos, na presente tese de doutorado apresentam-se dois algoritmos para Redes Oportunistas, os quais tomam suas decisões de roteamento e gerenciamento de recursos considerando os laços sociais entre os nós da rede. Para o problema do roteamento, apresenta-se o algoritmo *Friendship and Selfishness Forwarding* Quando surge uma oportunidade de contato, o algoritmo proposto primeiramente classifica os laços sociais entre o destinatário da mensagem e o nó candidato a receber a mensagem, doravante referido como *relay*. Posteriormente, utilizando funções logísticas, o algoritmo proposto avalia o egoísmo do nó *relay* para considerar os casos em que o nó receptor é egoísta seja porque seu dispositivo está com limitações de recursos, ou porque ele é racionalmente egoísta. Para o problema do gerenciamento de *buffer*, é introduzido o algoritmo denominado *Friendly-drop* (FDA), o qual toma suas decisões de encaminhamento/descarte de mensagens baseando-se nos relacionamentos sociais entre os nós. Quando o *buffer* dos nós está cheio, FDA prioriza o descarte de mensagens destinadas a usuários com quem a relação social é mais fraca. Por outro lado, quando os nós estão em contato, FDA prioriza o envio de mensagens destinadas a usuários com quem a relação social é mais forte.

Os resultados obtidos através do simulador *The ONE* mostram que, mesmo considerando o egoísmo dos nós no problema de disseminação de mensagens, o algoritmo proposto supera outros algoritmos bem conhecidos na literatura, aumentando a taxa

de entrega em até 20% e com a vantagem de precisar de um menor número de eventos de encaminhamento. Os resultados obtidos na presente tese de doutorado também demonstram que o algoritmo de gerenciamento de *buffer* pode se tornar uma importante chave para melhorar o desempenho da rede em cenários com nós egoístas.

Palavras-chave: Redes oportunistas, Aprendizagem de máquina, disseminação de mensagens.

Abstract

Opportunistic networks provide communication support in areas with overloaded cellular networks, and in scenarios where a fixed infrastructure is not available, as in remote and developing regions. A critical issue, which still requires a satisfactory solution, is the design of an efficient data delivery solution that considers delivery efficiency, delay, and cost. To tackle this problem, most researchers have used either the network state or node mobility as a forwarding criterion. Solutions based on social behaviour have recently being considered as a promising alternative.

Following the philosophy from this new category of protocols, in this work, we present the "FriendShip and Acquaintanceship Forwarding"(FSF), a routing algorithm that makes its routing decisions considering the social ties between the nodes and both the selfishness and the device resources levels of the candidate to message relay. When a contact opportunity arises, FSF first classifies the social ties between the message destination and the candidate to relay. Then, by using logistic functions, FSF assesses the relay node selfishness to consider those cases in which the relay node is socially selfish. To consider those cases in which the relay node does not accept to receive the message because its device has resource constraints at that moment, FSF looks at the resource levels of the relay node. Regarding the buffer management problems, in this work we introduced the algorithm Friendly-Drop (FDA) which takes its decisions based on the nodes social relationships. When the nodes buffer is with constraints, FDA drops the messages addressed to nodes with weak social relationship. On the other hand, if a pair of nodes are in contact, FDA sends the messages addressed to nodes having strong social relationships.

By using the ONE simulator to carry out trace-driven simulation experiments, we have found that even considering the selfishness on routing issues, our FSF algorithm outperforms previously proposed schemes, by increasing the delivery ratio up to 20% with the advantage it needs a lower number of forwarding events. We have also found that the buffer management algorithm can become an important key to improve network performance in scenarios with selfish nodes.

Keywords: Computer Networks, Machine learning, Routing algorithm.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Redes de comunicação de dados (RCD) tornaram-se nos últimos anos um item fundamental para o avanço do acesso à informação em todo o mundo. De acordo com [4], a Internet, principal exemplo de uma RCD, é o segundo meio de comunicação mais utilizado em todo o mundo, sendo superado somente pela televisão. Através do uso de RCDs, tornou-se possível o compartilhamento de informações e o desenvolvimento de diversas aplicações que facilitam serviços, auxiliam e oferecem entretenimento à população mundial. De acordo com [49], grande parte desse sucesso deve-se à operação dos protocolos da pilha TCP/IP, os quais são base da arquitetura da Internet.

Os protocolos que atuam na pilha TCP/IP necessitam, para o seu bom funcionamento, que determinadas condições típicas sejam atendidas, tais como: rota fim-a-fim na comunicação entre fonte e destino, baixa perda de pacotes, baixo atraso na troca de dados, entre outros [15]. No entanto, em diversos cenários, uma ou mais dessas premissas básicas para a comunicação não podem ser fornecidas, seja por questões estruturais ou até mesmo por falta de investimentos da iniciativa pública, ou privada. A ausência dessas premissas torna a operação desses protocolos inadequados e pouco robusta em tais cenários [12]. Exemplos de tais ambientes são redes de comunicações sem fio, comunicações entre dispositivos móveis, entre equipamentos com restrições de energia, comunicações rurais ou submarinas, além da Internet interplanetária [15]. Redes Tolerantes ao Atraso e Desconexões (*Delay and Disruption Tolerant Networks*(DTNs) [14] foi o nome dado a esse conjunto de ambientes que compartilham das características citadas acima. Esses cenários de rede vêm ganhando muita atenção de diversos pesquisadores nos últimos anos, visto que, dentre diversas questões existentes para que ocorra a comunicação nesses cenários, somente para a arquitetura base para as DTNs [14] é que já existe um consenso por parte da comunidade científica.

As Redes Oportunistas, doravante citadas pelo seu acrônico em inglês OppNet

(Opportunistic Networks), são um tipo especial de DTN em que os nós que compõem a rede são basicamente dispositivos móveis carregados por seres humanos [77]. Assim como nas demais DTNs, a disseminação de mensagens em OppNets torna-se possível por conta da utilização do mecanismo de armazena-carrega-encaminha [12]. Nesse mecanismo, da mesma maneira que em redes convencionais, quando um nó deseja enviar uma mensagem ocorre uma tentativa de comunicação com o nó destino. Se não existe um caminho fim a fim conectando esses nós, o remetente armazena a mensagem de maneira persistente para tentar seu envio em contatos futuros [39]. É interessante destacar que, para aumentar a probabilidade de entrega, um nó remetente pode compartilhar com outros nós a responsabilidade de entregar a mensagem ao seu destinatário, ou seja, ele encaminha a mensagem para outros nós que podem também repassá-la ao seu destino.

Esse mecanismo de comunicação, apesar de funcional, introduz algumas questões que precisam ser consideradas. Para esclarecer tais questões, considere os nós N1, N2 e N3, e uma mensagem M1 carregada por N1 cujo destinatário é N3. Considere ainda que em um dado momento, N1 está em contato com N2. Dentre as questões a serem consideradas no problema do roteamento, a primeira é: N1 deve encaminhar a mensagem M1 para N2? Essa primeira questão levantada trata-se do tradicional desafio de um algoritmo de roteamento em uma OppNet, que é decidir que mensagens serão encaminhadas em uma oportunidade de contato, de maneira a construir um caminho entre um par de nós, permitindo a troca de mensagens entre eles. Uma técnica eficiente para atingir esse objetivo é selecionar como *relay* aqueles nós candidatos que tem boas probabilidades de encontrar o destino em seus próximos contatos. É fácil perceber que quanto melhor essa seleção, maiores serão as chances de se obter sucesso na entrega de uma mensagem.

No exemplo acima, uma segunda questão que surge é: N2 deve obrigatoriamente aceitar o recebimento da mensagem M1? Essa segunda questão é importante e também é um dos desafios de um algoritmo de roteamento. Em cenários de redes reais, um nó pode autonomamente decidir se aceita ou não o recebimento de uma mensagem cujo destinatário é outro nó. Dessa forma, a cooperação do nó não é totalmente garantida [23]. Assim, se o algoritmo de roteamento decide encaminhar uma mensagem a um *relay* que é egoísta, ele pode se recusar a receber a mensagem caso o seu destinatário seja um nó desconhecido ou indesejado. Então, como proceder se o nó selecionado como *relay* não estiver disposto a carregar a mensagem? No que diz respeito ao egoísmo, de acordo com [91], os nós podem ser classificados como sociais ou individualmente egoístas. Basicamente, um nó classificado como individualmente egoísta não colabora com nenhum outro nó, o que significa que ele não aceita mensagens de outros nós

na rede. Por outro lado, um nó classificado como socialmente egoísta está disposto a retransmitir mensagens para os nós dentro da mesma comunidade, mas se recusa a armazenar/encaminhar mensagens para os nós que não fazem parte do seu ciclo social [42]. Portanto, esses nós somente retransmitem mensagens para nós com os quais possuem algum relacionamento social (amigos ou conhecidos). Assim, considerando essas suposições, é fácil perceber que um algoritmo de roteamento deve ser proposto incluindo a detecção de nós egoístas e também descobrindo o conjunto de nós que estão dispostos a colaborar com ele.

Por outro lado, nos últimos anos, alguns pesquisadores têm mostrado que o algoritmo de gerenciamento de *buffer* causa um grande impacto no processo de disseminação de mensagens em OppNets [16, 24]. O gerenciamento do *buffer* dos nós é uma tarefa decorrente dos frequentes eventos de *overflow* que ocorrem por conta do espaço limitado de memória secundária dos dispositivos carregados pelos nós da rede. Quando esses eventos ocorrem, automaticamente uma ou mais mensagens devem ser descartadas para que uma nova mensagem seja armazenada no *buffer* do nó. Essa escolha é importante para o desempenho da rede, visto que mensagens que têm uma alta probabilidade de serem entregues nos próximos contatos do nó podem ser descartadas. Dessa forma, o algoritmo de gerenciamento de *buffer* deve ser capaz de detectar quais são as mensagens com maior probabilidade de entrega para que sua manutenção no *buffer* seja garantida.

Outra característica essencial de um algoritmo de gerenciamento de *buffer* é detectar que mensagens devem ser enviadas primeiramente em uma oportunidade de contato. Essa escolha é fundamental para que em cada oportunidade de contato as mensagens de interesse sejam transmitidas integralmente. Isso ocorre porque a duração dos contatos entre os nós pode não ser suficiente para enviar e receber todas as mensagens que gostariam trocar entre si. Em ambos os casos, uma boa estratégia a ser seguida pelo algoritmo de gerenciamento de *buffer* é utilizar os relacionamentos sociais como critério para as decisões de descarte e envio de mensagens. Dessa forma, é razoável supor que, quando ocorre o evento do *buffer overflow*, as mensagens com maior probabilidade de serem entregues em contatos próximos são aquelas destinadas aos nós com quem o nó que toma a decisão tem um relacionamento social mais forte. Da mesma forma, quando em uma oportunidade de contato, é razoável supor que as mensagens mais interessantes a serem enviadas para um *relay* são aquelas destinadas a outros nós com quem ele tem uma relação social mais forte.

De acordo com as premissas citadas acima, uma proposta de algoritmo de roteamento para OppNets deveria ser composta necessariamente por um bom mecanismo capaz de determinar se um dado nó é um bom *relay* para uma dada mensagem, além

da capacidade de determinar se o nó é egoísta ou não. Além disso, uma interessante proposta de algoritmo de gerenciamento de *buffer* dos nós seria utilizar relacionamentos sociais como critério para as decisões de encaminhamento e descarte de mensagens.

1.1 Redes OppNet

As OppNets são um tipo especial de DTN cujos nós são basicamente dispositivos móveis carregados por seres humanos [77]. Para que a disseminação de mensagens ocorra nesse ambiente de rede, cada nó primeiramente armazena de maneira persistente as mensagens que deseja enviar em oportunidades de contato próximas. As oportunidades de contato surgem quando os nós, utilizando suas interfaces *Bluetooth* ou *Wi-fi* descobrem que ambos estão no mesmo raio de cobertura. Nesse momento, os nós podem então realizar os procedimentos de transmissão e recebimento de dados.

Evidentemente, o roteamento é um dos maiores desafios, principalmente por conta dos longos atrasos e frequentes desconexões existentes. Como mostrado na Seção 4.1, pesquisadores têm utilizado diversos critérios no processo de roteamento de mensagens, tais como a mobilidade dos nós e características globais da rede. A utilização da mobilidade como um critério para disseminação tem se mostrado interessante, principalmente porque as oportunidades de contato são criadas à medida que os nós movimentam-se pela rede. No entanto, a complexidade em encontrar padrões matemáticos que descrevam a mobilidade dos nós torna mais difícil a sua utilização. A questão da complexidade afeta também o uso de características globais da rede como critério para o roteamento de mensagens. O principal desafio é a estimação de determinados parâmetros, visto que a rede é formada por várias partes desconectadas, dificultando a correta estimativa de informações fundamentais para a tomada de decisões de envio de mensagens.

Mais recentemente, o uso de características sociais tem ganhado destaque nas pesquisas referentes a roteamento em OppNets. A principal motivação é o fato de características sociais serem menos voláteis que as demais utilizadas como critério de roteamento. Isso implica que, com o passar do tempo, existem mudanças menores nas estimativas realizadas, ou seja, se dois nós são considerados amigos, existe uma probabilidade alta de essa estimativa não mudar por um longo tempo. A utilização de características sociais como critério para o projeto de algoritmos em OppNets tem apresentado resultados bastante promissores [66, 6] e é um tema bastante recente, para o qual existem diversas lacunas a serem preenchidas na literatura. Especificamente no problema de disseminação de mensagens em OppNets, entre as características sociais mais utilizadas estão a popularidade e a centralidade [19, 20]. Apesar de contribuir

para degradar o desempenho da rede, o egoísmo entre os nós tem recebido bastante atenção por parte da comunidade científica, tais como em [31, 33, 35].

[6] Bulut et al. introduziram o uso da amizade entre os nós como um interessante critério para a disseminação de mensagens. A proposta é intuitiva: se dois nós são amigos, existe uma boa probabilidade de eles se reunirem em algum momento. O grande desafio na utilização desse relacionamento social como critério para disseminação de mensagens é a sua própria representação, ou seja, como serão detectadas as relações sociais existentes entre os nós que compõem a rede. Nos trabalhos encontrados na literatura [66, 6], os autores representaram matematicamente a amizade entre os nós utilizando evidências como a quantidade, a durabilidade, e a longevidade dos contatos entre os nós. Apesar dessa abordagem ser considerada interessante, a utilização de dados relacionados aos contatos dos nós para avaliar o relacionamento de amizade entre eles pode levar a cenários em que falsos positivos são identificados, e isso pode impactar na disseminação de mensagens em OppNets. Outro ponto a se destacar é que esses autores têm considerado apenas dois tipos de relacionamentos sociais entre os nós: amigos ou desconhecidos. No entanto, no mundo real existem diversos outros relacionamentos interessantes a serem considerados, como, por exemplo, as relações familiares, de trabalho, entre outras. O fato de as relações sociais serem compostas de características que sofrem muita variação dificulta a criação de um mecanismo matemático que as represente corretamente.

Por outro lado, na literatura para OppNets, observa-se que a maioria dos algoritmos de roteamento propostos considera que a participação dos nós na rede é garantida. No entanto, os nós podem ser egoístas e recusarem o recebimento de mensagens destinadas a nós com quem eles não têm uma determinada relação social. Na literatura, os autores em [30] introduziram a consideração do egoísmo em cenários de avaliação de algoritmos de roteamento. No entanto, diversas lacunas ainda são observadas na literatura, como, por exemplo, a ausência de mecanismos de detecção do comportamento egoísta dos nós associado às decisões tomadas pelo algoritmo de roteamento. É interessante destacar que esses fatores impactam diretamente nos resultados obtidos dentro dos cenários avaliados e a consideração de tais premissas torna os resultados obtidos mais confiáveis pelo fato de o cenário avaliado ter uma similaridade maior com um cenário de rede real.

1.2 Objetivos

A presente tese de doutorado tem como objetivo geral contribuir para melhorar o desempenho com relação a quantidade de mensagens entregue em cenários de rede do tipo OppNet. A partir desse objetivo, considerando as lacunas identificadas na literatura para redes do tipo OppNet, a presente tese de doutorado tem como objetivos específicos:

- Proposição, avaliação e análise de algoritmos que melhorem o roteamento de mensagens na rede.
- Proposição, avaliação e análise de algoritmos que melhorem o gerenciamento de recursos dos nós da rede, com foco principal no problema do gerenciamento de *buffer* dos nós.

1.3 Motivação

As pesquisas científicas sobre temas relacionados a OppNets têm apresentado resultados bastante promissores nos últimos anos. Além disso, um enorme atrativo nessa tecnologia é o baixo custo de implementação, algo que certamente é um ponto que pode favorecer o aumento da sua utilização. Apesar disso, ainda se percebe uma quantidade baixa de aplicações de tal tecnologia no mundo real. Os principais motivos para isso envolvem questões como a quantidade pequena de aplicações já desenvolvidas que atuem sob OppNets, o alto consumo de recursos dos dispositivos que compõem a rede, tais como bateria e memória secundária, ou mesmo a falta de motivação dos usuários e operadores em contribuir para a disseminação de mensagens na rede.

Na literatura para OppNets, encontra-se uma quantidade significativa de trabalhos relacionados a temas importantes para o funcionamento da rede. A maioria desses trabalhos divide o processo de disseminação de mensagens em OppNets em duas partes principais: o roteamento de mensagens e o gerenciamento de recursos dos dispositivos. Apesar de serem partes diferentes, ambas se complementam e é essencial que exista um equilíbrio para que o processo de disseminação de mensagens obtenha sucesso. A maioria dos trabalhos encontrados na literatura abordando esses temas realizam modelagens analíticas e simulações, porém, percebe-se que grande parte desses trabalhos desconsidera aspectos importantes do processo de disseminação de mensagens em redes reais, tais como a limitação de recursos dos dispositivos que compõe a rede, dificultando assim uma análise mais completa do desempenho em cenários de redes reais.

Dessa forma, é razoável supor que a tecnologia das redes OppNets pode ser utilizada em uma escala maior, caso os algoritmos de disseminação de mensagens se tornem mais eficientes e eficazes, consumindo menos recursos dos dispositivos, e respeitando as decisões dos usuários com relação a que mensagens devem ser armazenadas em seus respectivos dispositivos. Assim, o foco da presente tese de doutorado foi o desenvolvimento de algoritmos que considerem a maior parte desses aspectos no processo de disseminação de mensagens, tornando a análise de desempenho mais real, contribuindo para melhorar o desenvolvimento das OppNets.

1.4 Metodologia a ser utilizada

Para alcançar os objetivos traçados, foi realizada inicialmente uma profunda revisão bibliográfica do tema abordado no presente trabalho. Além disso, para propor e desenvolver os algoritmos propostos na presente tese de doutorado, realizou-se uma metodologia incremental. Primeiramente, foi proposto, testado, analisado e comparado uma proposta de algoritmo para o roteamento de mensagens em OppNets com nós egoístas. Posteriormente, foi proposto, testado, analisado e comparado uma proposta de algoritmo de gerenciamento de *buffer* para OppNets. Finalmente, realizou-se a integração de ambos os algoritmos, testando-se e analisando-se o desempenho da integração realizada.

Para realizar a validação dos algoritmos propostos no presente trabalho foi usado a técnica da simulação. A escolha pela técnica da simulação, invés de outras se dá por conta da enorme utilização dessa técnica em trabalhos relacionados a OppNets, além dos baixos custos na utilização desse recurso. O simulador utilizado foi o *The One Simulator* [22]. Esse simulador era a ferramenta mais utilizada no processo de proposição, validação e análise de novos algoritmos para OppNets. Trata-se de um programa de código aberto escrito na linguagem Java, fornecendo implementações que simulam itens como a mobilidade de nós em cenários de rede, Camada de Aplicação e de Rede.

1.5 Contribuições

As contribuições da presente tese de doutorado encontram-se elencadas a seguir:

- a elaboração de uma base de treinamento contendo dados relacionados aos relacionamentos sociais entre os nós. Esses dados foram retirados de um experimento

realizado em um ambiente de rede real. Essa base pode ser utilizada por diferentes pesquisadores como base de treinamento por algoritmos de aprendizagem de máquina. (Capítulo 3)

- a avaliação de algoritmos de aprendizagem de máquina supervisionados como uma ferramenta auxiliar para detecção dos relacionamentos sociais entre os nós da rede. (Capítulo 3)

- a elaboração de uma representação matemática da existência de um relacionamento social entre os nós da rede. (Capítulo 3)

- a elaboração de um mecanismo de detecção do egoísmo dos nós da rede. (Capítulo 3)

- a proposta de um algoritmo de roteamento que utilize os relacionamentos sociais entre os nós da rede como critério para a disseminação de mensagens, e possa detectar o egoísmo dos nós da rede. (Capítulo 4)

- a proposta de um algoritmo para o gerenciamento de *buffer* dos nós que utilize em suas decisões os relacionamentos sociais entre os nós da rede. (Capítulo 5)

- uma avaliação minuciosa dos algoritmos propostos. (Capítulos 4 e 5)

1.6 Organização do trabalho

O restante deste projeto de tese está organizado da seguinte forma:

- O Capítulo 1 introduz o trabalho, bem como apresenta a motivação e os objetivos a serem alcançados.

- O Capítulo 2 apresenta o estado da arte, apresentando os principais itens necessários para o desenvolvimento e entendimento da presente tese de doutorado.

- O Capítulo 3 apresenta as técnicas que serão utilizadas nos algoritmos propostos no presente trabalho.

- O Capítulo 4 apresenta uma proposta de algoritmo de roteamento que utiliza as técnicas para detecção de relacionamentos sociais entre os nós e do nível de egoísmo do nó candidato a *relay*.

- No Capítulo 5 uma proposta de algoritmo para o gerenciamento de *buffer* dos nós é descrita, bem como é apresentada uma análise dos resultados obtidos nas avaliações de desempenho realizadas.
- O Capítulo 6 apresenta uma análise mais aprofundada do desempenho do algoritmo proposto no Capítulo 4. O principal objetivo é identificar pontos de melhoria a serem implementados em novas versões do algoritmo.
- No Capítulo 7 são listadas as principais conclusões obtidas a partir das atividades realizadas na presente tese de doutorado.
- Finalmente, o Capítulo 8 sumariza as principais publicações decorrentes dos resultados obtidos na presente tese de doutorado.

Capítulo 2

Fundamentação Teórica

Para orientação e compreensão da aplicação da teoria envolvida na pesquisa desenvolvida no presente trabalho, é necessário o estudo das OppNets. Por se tratar de um tipo particular, as OppNets têm a mesma arquitetura e diversas características derivadas das DTNs. Dessa forma, nas seções subsequentes apresentam-se diversos conceitos básicos de DTN que são necessários para o entendimento da investigação realizada no presente trabalho.

2.1 Redes Tolerantes a Atrasos e Desconexões

2.1.1 Conceito

A Internet tornou-se um dos principais meios de comunicação no mundo. Através dessa 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 bases da arquitetura da Internet.

Segundo [49], 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 premissas, fato que torna a operação desses protocolos inadequados e pouco robusta em tais ambientes [12]. Convencionou-se chamar os ambientes que consideram estes aspectos por Redes Tolerantes ao Atraso e Desconexões (*Delay and Disruption Tolerant Networks* – DTNs) [14]. Exemplos de tais ambientes são redes de comunicações sem fio, co-

municações entre dispositivos móveis, comunicações entre dispositivos com restrições de energia, comunicações rurais, comunicações submarinas e comunicações interplanetárias [15]. As Figuras 2.1 e 2.2 exemplificam alguns desses ambientes no mundo real.

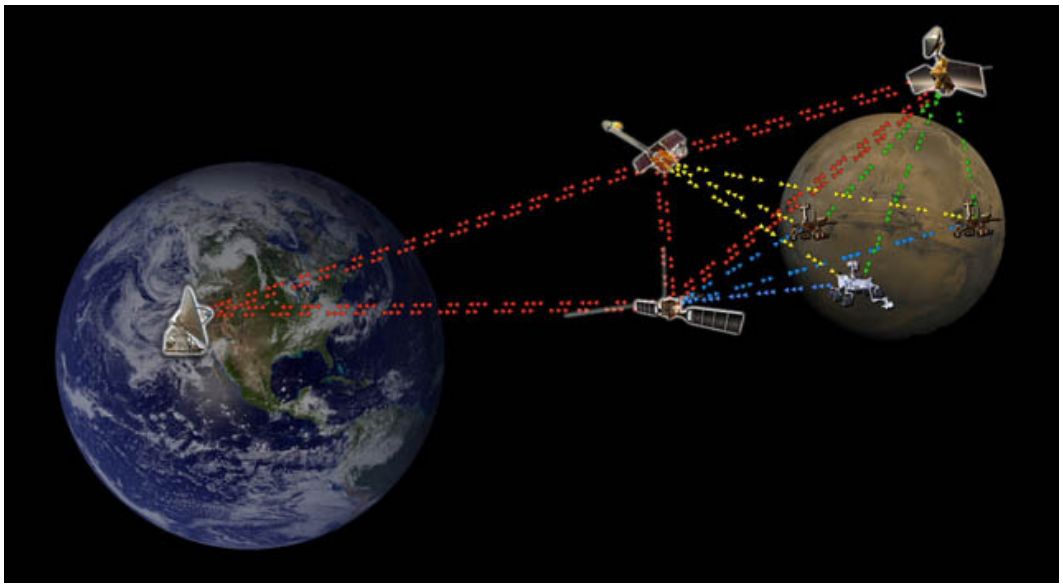


Figura 2.1: Exemplo 1: Internet Interplanetária. Nesta modalidade de Internet, os atrasos na comunicação podem ser da ordem de horas e até mesmo dias. Fonte:[1]

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

Os aspectos comuns a todas as DTNs dificultam principalmente a operação do

protocolo *Transmission Control Protocol* – TCP. O TCP é um protocolo orientado à conexão que garante confiabilidade na entrega de dados fim-a-fim [12]. 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.

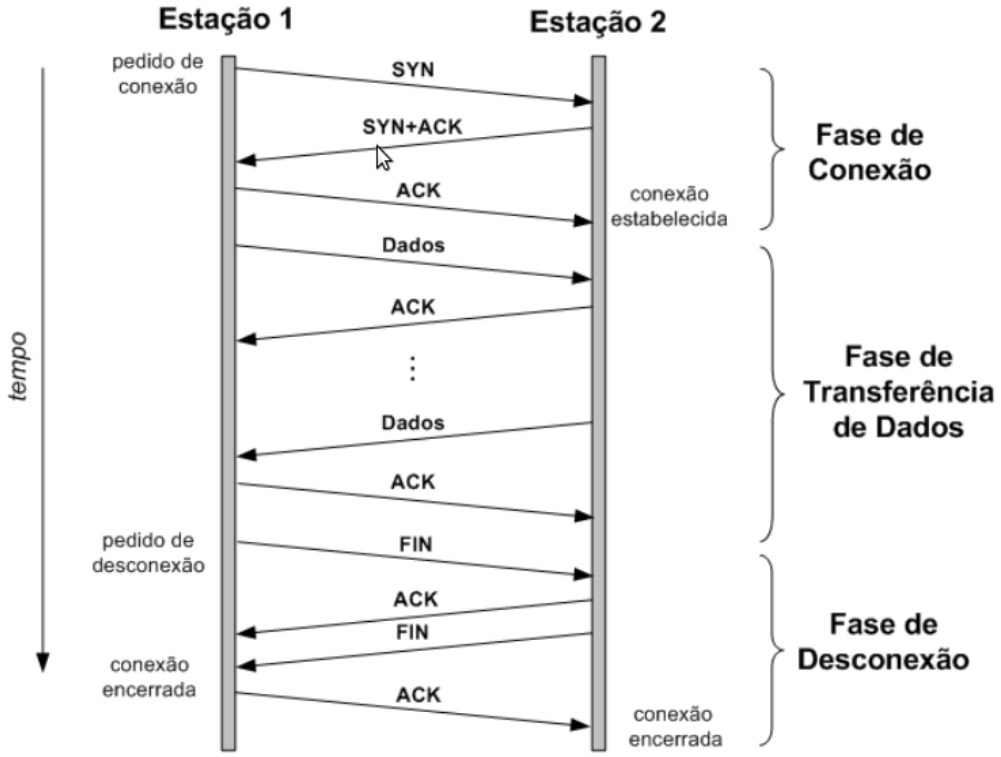


Figura 2.3: Funcionamento do protocolo TCP (Fonte: [12])

Segundo [53], 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 protocolo 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 a seguir.

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), denominada 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 [1]. 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 frequentes desconexões [11].

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 se deseja 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 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, as DTNs são classificadas como redes do tipo armazena-carrega-e-encaminha (*store-carry-*

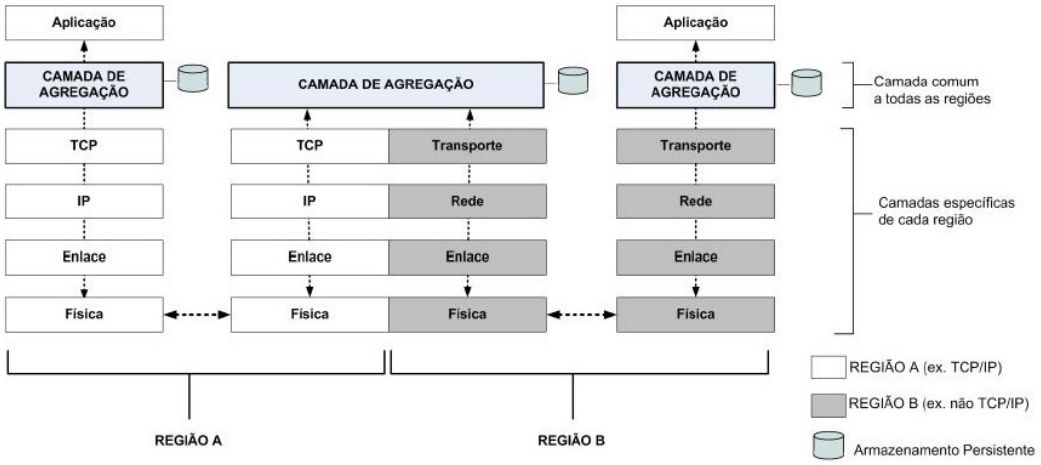


Figura 2.4: Camada de Agregação (fonte: [49])

and-forward), ou seja, primeiramente recebem totalmente a mensagem, armazenam persistentemente e somente depois enviam para o destino ou para nós intermediários [49].

Além da função principal que é possibilitar o armazenamento persistente, nessa camada também opera o protocolo de Agregação. O protocolo de Agregação é definido na RFC 5050 [56], 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 ADUs, os quais também são denominados como *bundle*, 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 seguir são apresentadas as categorias de contatos em DTNs.

2.1.3 Tipos de Contato em DTNs

Segundo [82], um contato em uma DTN corresponde a uma ocasião favorável à 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 categorias de contatos: persistentes, em demanda, agendados, oportunistas e previsíveis. A seguir uma breve descrição de cada

uma dessas categorias de contatos.

- persistente - segundo [46], 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 [49], 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 [49], programado. Um exemplo desta categoria 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. Considerando essa informação, os nós podem fazer uma previsão de que momento ocorrerá o contato com determinado nó da rede.

Para se avaliar o desempenho de uma DTN, são necessárias métricas que indiquem 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 [22], 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 - mede a quantidade de número de mensagens recebidas em razão do número de mensagens enviadas.
- Atraso de Entrega - mede o intervalo de tempo médio entre o envio e o recebimento das mensagens na rede.

- Ocupação da rede - mede a porcentagem média de ocupação de espaço de armazenamento dos nós da rede em um determinado tempo.
- 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.

Apesar do atual consenso da comunidade científica sobre a arquitetura utilizada nas DTNs [11], ainda existem diversos problemas nessa modalidade de comunicação que continuam em aberto e são alvos de diversas pesquisas no mundo todo. A seguir apresentam-se detalhes sobre os principais problemas relacionados a disseminação de mensagens em DTNs.

2.2 Aprendizagem de Máquina

Aprendizagem de máquina pode ser definida como qualquer mudança em um sistema que melhore o seu desempenho na segunda vez que ele repetir a mesma tarefa, ou outra tarefa da mesma população [63].

Existem diversos métodos de aprendizagem de máquina na literatura, os quais podem ser mais adequados para algumas classes específicas de problemas, por exemplo, quando em um problema tem-se uma determinada quantidade de informações disponíveis (base de treino), a preferência é utilizar métodos de aprendizagem de máquina que necessitem dessas informações anteriormente ao início de sua execução.

No presente trabalho, selecionou-se o algoritmo de aprendizagem de máquina supervisionado Naive-Bayes. A seguir, maiores detalhes sobre essa técnica.

2.2.1 Algoritmo Naive Bayes

Em um problema de classificação em aprendizagem de máquina, o classificador Naive Bayes usa o Teorema de Bayes para calcular as probabilidades necessárias para classificar uma nova instância. A partir de uma perspectiva de aprendizado de máquina, pode-se definir o teorema de Bayes da seguinte forma: dada uma instância desconhecida $A = (a_1, a_2, a_3..a_n)$, onde $a_i (i = 1, \ldots n)$ são os valores dos atributos de uma instância, deseja-se prever a qual classe A pertence.

Segundo o teorema de Bayes, a probabilidade de escolher uma classe dada uma amostra é dada por:

$$P(Classe|A) = \frac{P(A|Classe) \times P(Classe)}{P(A)} \tag{2.1}$$

Pode-se reescrever a Equação 2.1 em termos dos atributos da instância A:

$$P(Classe|a_1, .., a_n) = \frac{P(Classe|a_1, ..., a_n) \times P(Classe)}{P(a_1, ..., a_n)} \tag{2.2}$$

A definição de classe da instância A envolve o cálculo da probabilidade de todas as classes possíveis, dado um atributo específico. A classe definida é aquela com a maior probabilidade. Do ponto de vista estatístico, isso equivale a maximizar P (Classe | a_1, Classe |a_2, ..., Classe | a_n). O denominador na Equação 2.2 é constante, o que leva à seguinte simplificação:

$$P(Classe|A) = argmax \prod_i (P(a_1, ..., a_n|Classe) \times P(Classe) \tag{2.3}$$

Para as tarefas de aprendizagem de máquina a serem realizadas no presente trabalho, escolheu-se esse classificador levando em consideração resultados obtidos em trabalhos anteriores, nos quais Naive Bayes obteve desempenho promissor [69, 70].

Capítulo 3

Arquitetura utilizada

Neste Capítulo são apresentados em detalhes o cenário e as técnicas propostas que são utilizadas pelos algoritmos propostos na presente tese de doutorado. Primeiramente apresenta-se o cenário para o qual se aplicam estes algoritmos. Posteriormente, são dados detalhes das técnicas utilizadas para a classificação dos relacionamentos sociais e a detecção do egoísmo dos nós.

3.1 Cenário utilizado

Os algoritmos propostos no presente trabalho consideram que (i) os nós podem comportar-se de maneira egoísta, e (ii) as decisões de roteamento e gerenciamento de recursos são baseadas nas relações sociais entre os nós da rede. A seguir, primeiramente apresenta-se o modelo de rede e como é detectado o egoísmo dos nós. Então, apresentam-se as técnicas utilizadas para determinar as relações sociais entre os nós da rede.

3.1.1 Modelo de rede

O modelo de rede consiste em um conjunto de N nós, que são divididos em três grupos:

- I – contém os nós que se comportam como *individualmente egoístas*.
- S – contém os nós que se comportam como *socialmente egoístas*.
- C – contém os nós que se comportam como *não egoístas*.

Dessa forma, a quantidade de nós na rede é dada por |N| = |I| + |S| + |C|. Considera-se ainda que os nós não são mal-intencionados, ou seja, eles não enviam informações falsas pela rede (informações que impactam seu próprio nível de egoísmo).

3.2 Detalhamento da arquitetura utilizada

A arquitetura utilizada no presente trabalho é composta por dois mecanismos responsáveis por detectar os relacionamentos sociais e o egoísmo dos nós da rede: o *sistema classificador das relações sociais* (SCRS) e o *sistema de detecção de egoísmo* (SDE).

- **SCRS** – o principal objetivo deste sistema é atualizar as informações dos relacionamentos sociais entre cada par de nós na rede. No presente trabalho, os relacionamentos sociais podem ser classificados em três categorias: *amigos*, *conhecidos*, e *desconhecidos*. Um nó armazena informações sobre seus laços sociais em duas listas diferentes: *Meus Laços Sociais* (MLS), que armazena os seus próprios laços sociais, e *Laços Sociais Globais* (LSG) que armazena os relacionamentos entre os outros nós da rede.

  Para atualizar a lista *MLS*, cada nó possui um banco de dados contendo informações sobre seus encontros com outros nós da rede. O banco de dados que pertence ao nó n_i é composto por $n - 1$ linhas, onde a linha n_k armazena dados relacionados ao histórico de encontros entre os nós n_i e n_k. Inicialmente, esse banco de dados não possui informações armazenadas. A cada oportunidade de contato, ambos os nós em contato atualizam, em seus próprios bancos de dados, as informações relacionadas aos encontros entre si. Com base nessas informações, o SCRS pode classificar o relacionamento social com os demais participantes da rede. Essa classificação é realizada pela primeira vez quando é alcançado um dado tempo T configurado previamente. A cada novo intervalo de tempo T é realizada uma nova classificação.

  A lista LSG é atualizada em todas as oportunidades de contato que ocorrem após a primeira vez que o tempo T é alcançado. Dessa forma, em uma oportunidade de contato, os nós n_i e n_k primeiramente compartilham informações sobre seus próprios laços sociais (lista MLS). Baseando-se nessa informação, eles atualizam então a sua lista LSG. Utilizando essas listas, os nós da rede podem obter informações dos relacionamentos sociais entre os demais nós da rede.

- **SDE** – o principal objetivo deste sistema é atualizar a reputação do nó com relação ao egoísmo. Neste trabalho, com relação ao egoísmo, um nó pode ser

classificado como:

- **individualmente egoísta** - de acordo com [90], o nó individualmente egoísta é aquele que não está interessado em receber mensagens para outros nós; apenas deseja repassar suas mensagens a alguns destinatários. No presente trabalho, considera-se que os nós individualmente egoístas não aceitam receber mensagens.
- **socialmente egoísta** - pode ser considerado como socialmente egoísta o indivíduo que participa apenas parcialmente das ações de transmissão e retransmissão de mensagens na rede [90]. São aqueles nós que desejam armazenar e carregar mensagens destinadas a outros nós com quem ele tem uma relação social forte. No presente trabalho, esses nós retransmitem mensagens somente para aqueles com quem eles têm uma determinada relação de amizade.
- **não egoísta** - são aqueles nós que aceitam retransmitir mensagens para todos os outros nós.

Inicialmente, os nós têm a mesma reputação e nenhum conhecimento sobre qualquer informação da rede. A avaliação da reputação de um nó é realizada considerando o seu comportamento no passado ao ser selecionado como um nó *relay*. Nessa situação, no presente trabalho, um nó pode comportar-se de três maneiras: (i) sempre aceita receber a mensagem, (ii) não aceita receber a mensagem, (iii) aceita receber a mensagem somente se o seu dispositivo não tem restrições de recursos. Em todas as decisões, o nó atualiza sua própria reputação. O SDE é dividido em duas partes: um esquema de detecção, que é implementado como um sistema de *watchdogs* colaborativo baseado em [17], e um sistema de reputação que é baseado em uma função logística. A seguir, mais detalhes serão fornecidos sobre ambos os sistemas.

3.3 Sistema de avaliação do egoísmo dos nós

A Figura 3.1 mostra a interação entre os componentes do sistema de avaliação do egoísmo. Como citado anteriormente, o mecanismo de detecção é implementado como um sistema de *watchdog* colaborativo. De acordo com [40], esses sistemas são componentes instalados em cada nó da rede, os quais são responsáveis por ouvir a comunicação de dados entre os nós para decidir sobre o egoísmo do vizinho. Neste trabalho, utiliza-se um sistema de *watchdog* colaborativo baseado em [17].

Basicamente, quando um nó vizinho exibe um comportamento egoísta, o sistema de detecção tende a retornar uma detecção positiva. Caso contrário, ele avalia o nó como não egoísta. Apesar da simplicidade, esses sistemas de detecção apresentam resultados muito interessantes na detecção de comportamentos egoístas. No entanto, suas decisões podem ser afetadas pela existência de *outliers*. De acordo com [32], *outliers* são normalmente definidos como pontos de dados que têm uma diferença significativa em relação ao restante dos dados de acordo com uma determinada medida. Para diminuir o impacto de *outliers*, neste trabalho utiliza-se um sistema de reputação baseado em [64].

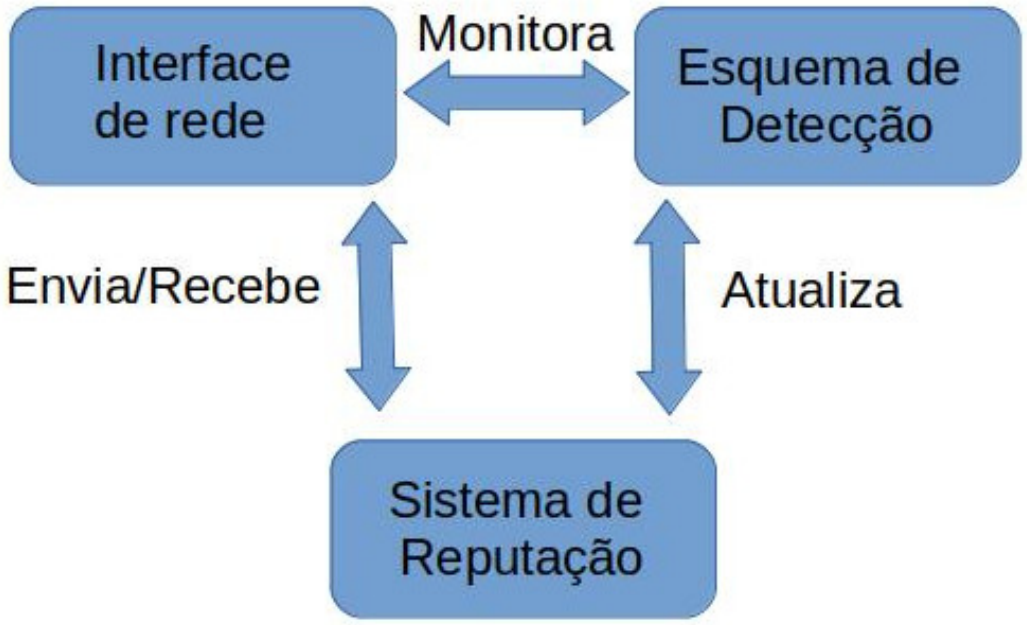


Figura 3.1: O sistema de avaliação do egoísmo executando em cada nó na rede.

Nesse sistema de reputação, a cada contato, uma avaliação é realizada, mesmo quando os nós são não egoístas. Basicamente, um nó A pode obter informações de um vizinho B nas seguintes situações:

Contato egoísta

O nó A, usando seu mecanismo de monitoramento durante um contato, detecta que um nó vizinho B é egoísta.

Contato não egoísta

Ambos os nós A e B não são egoístas. Assim, eles compartilham informações sobre o egoísmo dos demais nós da rede que eles conhecem.

A cada vez que o esquema de detecção coletar uma atualização sobre um vizinho, o sistema de reputação é atualizado. A reputação do nó é definida considerando a probabilidade de um nó ser mais cooperativo do que outros nós. Essa probabilidade é baseada em funções logísticas, como mostrado na Equação (3.1)[64]:

$$P_{coop}(A,B) = \frac{1}{1+10^{\frac{R_B - R_A}{F_d}}} \tag{3.1}$$

onde $P_{coop}(A,B)$ descreve a probabilidade do nó A ser mais cooperativo do que o nó B, R_A e R_B são respectivamente a reputação dos nós A e B, e F_d é um fator de significância para enfatizar a diferença entre a reputação dos nós A e B. A atualização da reputação do nó é definida como um processo de duas etapas para cada par de contatos (A,B): (i) verificar no passado o comportamento do nó B, e (ii) atualizar a reputação do nó B. O processo de atualização da reputação funciona da seguinte maneira:

Reputação média (RM)

Em cada contato, o nó calcula sua reputação média usando a Equação 3.2, a média aritmética da reputação dos vizinhos no momento de contato com o nó B.

$$RM(A) = \frac{\sum_{\forall k \in vizinhos_A - \{B\}} R_k}{|vizinhos_A| - 1} \tag{3.2}$$

Atualização da reputação

Quando um nó egoísta é detectado, o nó atualiza sua reputação usando a Equação 3.3, onde $D(B) \in \{-1, 1\}$ é 0 quando B é detectado como egoísta, e 1 em caso contrário. δ é o peso de cada nova observação.

$$R_B{}' = R_B + RF(D(B)) - P_{coop}(A,B)) \tag{3.3}$$

Usando esse modelo de reputação, cada nó pode inferir o comportamento de um nó vizinho com base em seu valor de reputação.

Além das decisões tomadas pelo sistema de reputação, no presente trabalho, um nó também pode comportar-se de maneira egoísta em situações de restrições de recursos. O objetivo é considerar os casos em que um nó classificado como socialmente egoísta ou não egoísta pode se recusar a receber uma mensagem porque naquele instante seu dispositivo se encontra com restrições de recursos. Para realizar essa avaliação, no presente trabalho, levou-se em consideração os seguintes recursos dos dispositivos: a memória secundária e o nível de energia.

Para simulação, criaram-se dois *thresholds*: α, que representa o nível de energia, e β que representa o espaço de memória secundária dos dispositivos. Quando surge uma oportunidade de contato, se o dispositivo do nó *relay* tiver um nível de energia menor que α ou se o espaço de memória secundário utilizado for maior que β, o nó recusará o recebimento da mensagem, embora seja socialmente egoísta ou não egoísta.

Para se decidir quais valores devem ser utilizados para esses *thresholds*, realizou-se uma avaliação utilizando várias combinações de α e β. Os resultados obtidos mostram que, a partir de valores menores que $\alpha = 30\%$ e $\beta = 70\%$, os resultados são muito semelhantes. Dessa forma, as simulações realizadas no presente trabalho consideram esses dois valores para esses *thresholds*.

3.4 Sistema de classificação dos relacionamentos sociais

No presente trabalho propõe-se um sistema para classificação dos relacionamentos sociais entre os nós da rede. O sistema proposto divide os relacionamentos sociais de um nó em três grupos: *amigos*, *conhecidos*, *desconhecidos*. A principal diferença entre os grupos de *amigos* e *conhecidos* é que amigos são uma relação social mais forte do que os *conhecidos* de um nó. Por exemplo, em uma sala de aula de uma universidade, alguns nós podem ser amigos de um nó n_x. Em geral, esses nós são aqueles com os quais n_x tem um relacionamento próximo. Na maioria das vezes, eles também se encontram fora da universidade. No entanto, existem os nós com os quais n_x não é muito próximo, mas eles estão frequentemente em contato. Por exemplo, para ir à universidade, n_x geralmente pega o mesmo ônibus e encontra o mesmo motorista (e outras pessoas) todos os dias. Apesar de não serem amigos, eles têm contatos regulares. Também vale a pena mencionar que as mensagens endereçadas aos amigos dos nós têm a prioridade mais alta do que aquelas endereçadas aos conhecidos, isto é, aquele nó priorizaria encaminhar e receber mensagens destinadas a seus amigos primeiro.

O sistema classificador de relacionamentos sociais é dividido em dois mecanismos:

- **Mecanismo de detecção de conhecidos** – é o mecanismo responsável por descobrir o grupo de conhecidos dos nós. Para alcançar este objetivo, propõe-se uma métrica que é uma extensão da métrica proposta por Li et al. [28].

- **Mecanismo de detecção de amigos** – é o mecanismo responsável por descobrir o grupo de amigos dos nós. Esse mecanismo considera dois cenários: (i) a infor-

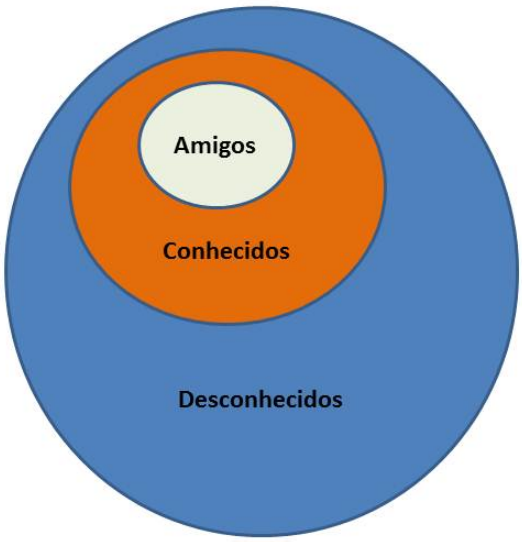


Figura 3.2: Categorias de relacionamento considerados no presente trabalho.

mação sobre os relacionamentos sociais entre os nós é conhecida (*self-reported*), e (ii) a informação sobre os relacionamentos sociais entre os nós não é conhecida.

A Figura 3.3 apresenta o ciclo realizado para realização dos procedimentos que constroem os grupos de conhecidos e amigos de cada nó na rede.

3.4.1 Mecanismo de detecção de conhecidos

Para descobrir o grupo de nós conhecidos de um nó, introduz-se uma métrica denominada *Nodes Acquaintanceship Metric* (NAM). A métrica NAM amplia a métrica proposta por Li et al. [28]. Seu principal objetivo é considerar a rotina diária das pessoas para descobrir quais nós pertencem ao grupo de nós conhecidos de um dado nó. É amplamente conhecido que os padrões de mobilidade das pessoas pode ser muito útil para determinar quando um par de nós se encontrará no futuro [51, 19]. A métrica NAM tenta tirar proveito dessa premissa, observando a similaridade dos contatos entre um par de nós.

Para o melhor entendimento do leitor, considere o exemplo mostrado na Figura 3.4. Nessa figura, pode-se ver o histórico de encontros entre o par de nós (id

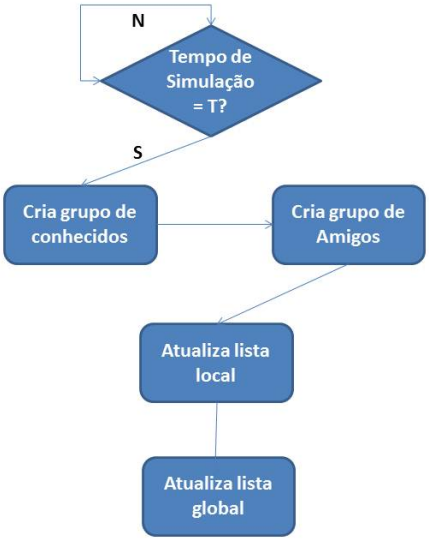


Figura 3.3: O sistema de classificação dos relacionamentos sociais executado por cada nó da rede.

13 e 22) participantes de um experimento real intitulado Sassy em quatro semanas diferentes [5]. De acordo com as informações disponibilizadas no projeto, os participantes declararam que não são amigos. Apesar disso, como é demonstrado na Figura 3.4, eles se encontram com frequência. Vale ressaltar que o histórico de contatos entre eles em cada semana tem valores semelhantes para algumas características, como a duração, frequência (cinco dias por semana), o tempo entre os contatos, entre outros. Com base nesse histórico de encontros, é razoável supor que há uma boa chance de eles estarem em contato novamente na semana seguinte.

No mundo real, é fácil perceber que esse comportamento geralmente ocorre nas rotinas diárias das pessoas. Por exemplo, esse comportamento pode ser aplicado a colegas no trabalho ou de curso da universidade, ou ainda pessoas compartilhando o ônibus, ou o metrô todos os dias, entre outros.

Com base nessas suposições, a métrica NAM considera que um par de nós x e y serão conhecidos um do outro se, e somente se, o histórico de contatos entre eles nos momentos T_i e T_{i+1} são semelhantes. Dessa forma, NAM utiliza o histórico de encontros entre x e y. Para medir a similaridade entre os contatos de um par de nós, NAM considera as seguintes características: quantidade de contatos (QC), duração do contato (DC), tempo entre contatos (TeC), e a hora mais comum de contato (HMCC). Essas características são estimadas da seguinte forma:

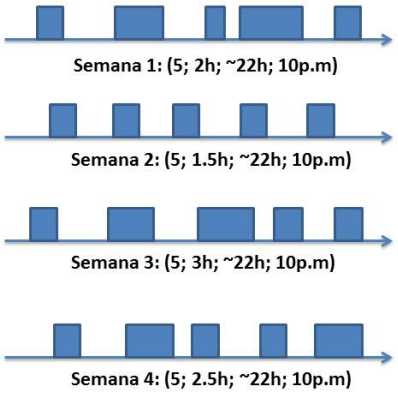


Figura 3.4: Histórico de encontros entre os nós 13 e 22 no experimento real Sassy. De acordo com os próprios participantes, eles não são amigos. No entanto, como mostrado acima, frequentemente eles estão em contato.

$$QC\ (x, y) = n \tag{3.4}$$

onde n é o número de encontros entre os nós (x, y) até um intervalo de tempo T,

$$DC\ (xy) = \frac{\sum_{i=1}^{n}(t_{fim(i)} - t_{incio(i)})}{n} \tag{3.5}$$

onde $t_{incio(i)}$ e $t_{fim(i)}$ são o horário de início e o horário de término do contato i entre os nós x e y,

$$TeC\ (xy) = \frac{\sum_{i=1}^{n}(t_{incio(i+1)} - t_{fim(i)})}{n} \tag{3.6}$$

onde $t_{incio(i+1)}$ é a hora de início do próximo contato e $t_{fim(i)}$ é a hora de término do contato anterior. De modo a diminuir o impacto da variabilidade dos dados, tanto para a duração do contato (DC) quanto para o tempo entre os contatos (TeC), NAM considera o desvio padrão dos dados coletados. A última característica usada pela métrica NAM é a *hora mais comum de contato*. Para essa métrica NAM armazena o horário em que os contatos com cada nó em específico ocorre. Em seguida, NAM usa o procedimento de moda estatística para determinar o horário mais comum dos contatos entre n_x e n_y.

Finalmente, para determinar a similaridade dos contatos entre n_x e n_y, NAM

utiliza uma métrica denominada similaridade cosseno [44]. Sejam u_i e v_i os vetores que contêm o histórico de contatos, nos momentos T_i e T_{i+1}, respectivamente. NAM calcula a similaridade entre u_i e v_i da seguinte forma:

$$similaridade_{u_i,v_i} = \cos(\theta) = \frac{u_i \times v_i}{||u_i|| \times ||v_i||} \tag{3.7}$$

onde $||u_i||$ e $||v_i||$ representam o produto escalar dos vetores u_i e v_i. Para determinar se um par de nós é conhecido um do outro, utiliza-se um *threshold* γ cujo valor é igual a 0,7. Se o valor da Equação 3.7 para os nós n_x e n_y for diferente de zero, e maior ou igual a γ, então eles serão considerados conhecidos um do outro.

Por exemplo, considerando o histórico de encontros mostrados na Figura 3.4 e os dados das semanas 1 (u_i = (5, 2, 22, 10)) e 2 (v_i = (5, 1,5, 22, 10)), o sistema de detecção de conhecidos utilizará os seguintes passos:

- Calcular $u_i \times v_i = (5 \times 5) + (2 \times 1,5) + (22 \times 22) + (10 \times 10) = 611$
- Calcular $||u_i|| = ((5^2) + (2^2) + (22^2) + (10^2))^{0,5} = 24,75$
- Calcular $||v_i|| = ((5^2) + (1,5^2) + (22^2) + (10^2))^{0,5} = 24,71$
- Calcular $\cos(\theta)$ de acordo com a Eq. 3.7 = $\frac{611}{||24,75|| \times ||24,71||} = 0,99$

Assim, neste exemplo, os nós $||u_i||$ e $||v_i||$ são considerados conhecidos um do outro. Finalmente, neste trabalho, a diferença entre T_i e T_{i+1} depende do cenário. A Tabela 3.2 inclui os valores de T usados nos diferentes cenários avaliados neste trabalho.

3.4.2 Sistema de detecção de amigos

Depois de descobrir os nós pertencentes ao grupo de conhecidos de um nó, o sistema de detecção de amigos verifica quais dos nós conhecidos tem uma relação ainda mais próxima (amigos). Para alcançar este objetivo, este sistema considera duas situações: (i) a informação sobre os relacionamentos sociais entre os nós está disponível (*self-reported*), e (ii) a informação não está disponível.

Na situação (i), cada nó apenas carrega essas informações em sua lista *MLS*. Essas informações podem ser encontradas em alguns conjuntos de dados derivados de experimentos reais. Por exemplo, no experimento *Reality Mining* [52], os autores forneceram à comunidade científica diversas informações dos usuários, incluindo a amizade reportada entre eles. A Figura 3.5 mostra a informação de amizade disponível no cenário Sassy [5].

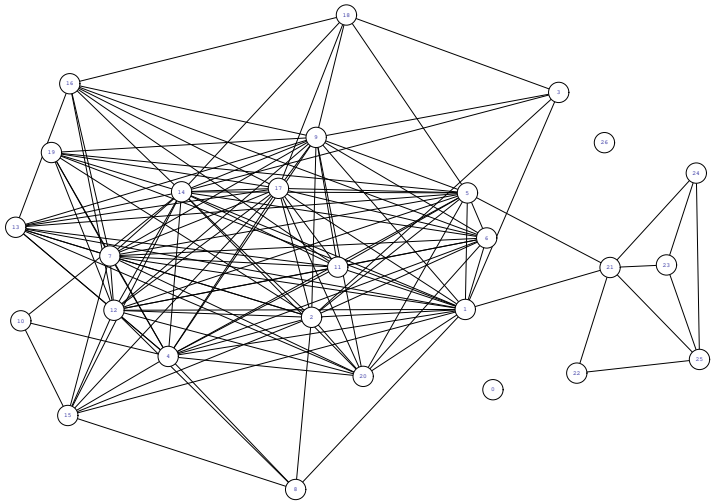


Figura 3.5: Grafo de amizade reportada pelos participantes do cenário Sassy.

Na situação (ii), o sistema de detecção de amizades usa um algoritmo de aprendizagem de máquina para descobrir a existência de amizade entre um par de nós na rede. Este esquema é composto de um algoritmo classificador e um banco de dados usado para treinar o classificador.

Para construir o banco de dados usado para treinar e testar o classificador Naive Bayes, utilizam-se os dados derivados do experimento de pesquisa do MIT [52]. Nesse experimento, os dados foram obtidos de um teste com dispositivos móveis equipados com uma interface *Bluetooth*. O experimento coletou dados de 97 telefones celulares ao longo de seis meses. Pesquisadores têm acesso às informações relacionadas à conectividade, proximidade, localização e atividades desses usuários. Além disso, os participantes foram convidados a informar o seu relacionamento com os demais participantes do experimento através da pergunta "Essa pessoa faz parte de seu círculo íntimo de amigos?". Baseando-se nessa informação, construiu-se uma base de dados que serve como treinamento do algoritmo Naive Bayes.

A base de dados de treinamento construída é composta pelos seguintes atributos e seus possíveis valores:

- **Número de encontros** (NE) - representa quantas vezes dois nós estiveram dentro do raio de alcance um do outro. Valores possíveis: baixo, médio e alto.

- **Duração média do contato** (DMC) - representa o tempo médio que dois nós permanecem dentro da área de cobertura um do outro. Valores possíveis: baixo, médio e alto.

- **Tempo médio entre os contatos** (TMEC) - representa o tempo médio entre o último contato e o novo contato. Valores possíveis: baixo, médio e alto.

- **Encontros fora da universidade** (EFU) - indica se um par de nós se encontrou fora da universidade. Valores possíveis: sim (se encontram fora da universidade), não (não se encontram fora da universidade).

- **Conhecidos em comum** (CC) - indica se um par de nós têm conhecidos em comum. Valores possíveis: sim (eles têm conhecidos em comum), não (eles não têm conhecidos em comum).

- **Relacionamento social** - representa a amizade reportada pelos participantes do experimento. Em uma perspectiva de aprendizado de máquina, é a classe. Valores possíveis: fortes (são amigos) e fracos (não são amigos).

Nesse banco de dados, os valores para os atributos Número de encontros (NE), Duração média do contato (DMC) e Tempo médio entre os contatos (TMEC) correspondem ao histórico de encontros entre cada par de nós do experimento do MIT no período que compreende 0 a T segundos. Para realizar a categorização dos valores para cada um dos atributos acima (baixo, médio, alto), utiliza-se N-percentis, onde N é a quantidade de categorias de cada atributo. Por exemplo, considere os valores de NE do experimento MIT representados por valores numéricos (0, 1, 2, ..., 50). Como se dividiu o atributo NE em três categorias, utilizaram-se então percentis (25% e 75%). Dessa forma, a categoria *NE=baixa* inclui 25% das ocorrências de valores de NE mais baixos, podendo compreender, por exemplo, os valores de 0 até 4, enquanto *NE = médio* entre 5 e 15. A partir de 16 encontros, considera-se *NE = alto*.

Escolheram-se os atributos NE, DMC e TMEC com base em outros trabalhos encontrados na literatura [6, 69]. Escolheram-se os atributos CC e EFU porque se acredita que eles são uma boa evidência da existência de uma relação social mais forte entre um par de nós. A Tabela 3.1 mostra um exemplo de algumas tuplas usadas neste trabalho pelo algoritmo Naive Bayes como dados de treinamento.

3.4.3 Exemplo de classificação Naive Bayes

Para melhor entendimento do leitor, considere o seguinte exemplo. Seja o banco de dados usado como treinamento do classificador Naive Bayes na Tabela 3.1, e considere que quando o tempo T mencionado acima é alcançado, o histórico coletado de encontros entre um par de nós *x,y)* é representado pela instância *A = (NE = alta, DMC = alta,*

Tabela 3.1: Um exemplo da base de dados de treinamento utilizada no presente trabalho.

NE	DMC	TMEC	EFU	CC	Relacionamento social
baixa	alta	media	sim	não	fraco
baixa	alta	alta	não	sim	fraco
media	media	media	não	sim	fraco
alta	media	baixa	sim	sim	fraco
alta	baixa	baixa	não	não	forte
alta	baixa	media	não	sim	forte
media	alta	baixa	sim	não	fraco
baixa	media	baixa	sim	não	fraco
alta	media	media	não	sim	forte
media	baixa	alta	não	não	fraco
media	alta	alta	não	sim	forte
alta	alta	alta	sim	sim	fraco
baixa	alta	media	sim	não	forte
media	media	baixa	não	não	fraco
baixa	baixa	baixa	não	sim	fraco

TMEC = baixa, EFU = sim, CC = não, relacionamento social?). Para responder se x e y têm uma relação social forte ou não, Naive Bayes utiliza os seguintes passos:

Passo 1 - Determina a probabilidade de ocorrência de cada classe. Trata-se da probabilidade $P(Classe)$ mostrada na Eq. 2.1. Basicamente, é o resultado da divisão do número de casos da classe X pelo número total de tuplas no banco de dados representado na Tabela 3.1:

$P(Classe = forte) = \frac{5}{15} = 0.34$

$P(Classe = fraco) = \frac{10}{15} = 0.66$

Passo 2 - Determina a probabilidade dos atributos da instância em questão sobre todas as classes possíveis. É a parte $P(Classe|a_1, ..., a_n)$ mostrada na Eq. 2.2:

$P(Classe = forte|NE = alto) = \frac{3}{5} = 0.6$

$P(Classe = fraco|NE = alto) = \frac{2}{10} = 0.2$

$P(Classe = forte|TMEC = alto) = \frac{2}{5} = 0.4$

$P(Classe = fraco|TMEC = alto) = \frac{4}{10} = 0.4$

$P(Classe = forte|ATBC = baixo) = \frac{1}{5} = 0.2$

$P(Classe = fraco|ATBC = baixo) = \frac{5}{10} = 0.5$

$P(Classe = forte|EFU = no) = \frac{4}{5} = 0.8$

$P(Classe = fraco|EFU = no) = \frac{5}{10} = 0.5$

$P(Classe = forte|CC = no) = \frac{2}{5} = 0.4$

$P(Classe = fraco|CC = no) = \frac{5}{10} = 0.5$

Passo 3 - Determina a probabilidade de cada classe com base na probabilidade da instância. É o resultado da Eq. 2.3:

$P(Classe = forte|A) = (0.6 \times 0.4 \times 0.2 \times 0.8 \times 0.4 \times 0.34) = 0.0005$

$P(Classe = fraco|A) = (0.2 \times 0.4 \times 0.5 \times 0.5 \times 0.5 \times 0.66) = 0.0066$

Neste exemplo, e de acordo com as probabilidades calculadas, o relacionamento social entre os nós x e y seria classificado como *fraco*, ou seja, os nós são desconhecidos um do outro.

3.5 Mobilidade dos nós

Para simular a mobilidade do nó, escolheram-se os seguintes traces de mobilidade real:

- **Cambridge** [3] –o trace de *Cambridge* recebeu este nome por ter sido gerado na Universidade de *Cambridge* com a contribuição 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 coletados dados de 19 estudantes do laboratório de computação equipados por 54 dispositivos utilizados para realizar as coletas que geraram o trace. O trace tem a duração de 11 dias.

- ***Reality*** [52] – o trace *Reality* é derivado 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.

- **NCCU** [78] – esse trace foi derivado de um experimento real realizado na Universidade Nacional de Chengchi. Nesse experimento, dispositivos móveis foram disponibilizados para 115 estudantes. Os registros de suas atividades relacionadas a mobilidade e troca de mensagens foram coletadas durante 15 dias. Tais informações foram utilizadas para criação do trace de mobilidade, algoritmos de roteamento, entre outros.

- **Sassy** [5] – esse trace foi derivado de um experimento real realizado na Universidade de St. Andrews. Neste experimento, os autores distribuíram 27 dispositivos para estudantes da universidade, totalizando um período de coleta correspondente a 79 dias.

A Tabela 3.2 lista alguns parâmetros descrevendo os traces usado nas simulações.

Tabela 3.2: Parâmetros dos traces utilizados nas simulações.

Trace	*Cambridge*	*Reality*	*NCCU*	*Sassy*
Dispositivo	iMotes	Smartphone	Smartphone	T-mote
Interface de rede	*Bluetooth*	*Bluetooth*	*Wi-fi*/*Bluetooth*	Sensor
Quantidade de nós	54	97	115	27
Duração do trace (dias)	11	246	15	79
Número de encontros (aprox.)	10.873	54.667	81.115	35.274
Tempo T (dias)	3	7	3	7

3.6 Algoritmos usados para comparação

Por uma questão de comparação, selecionaram-se os seguintes algoritmos de roteamento:

- **Epidêmico [10]** – o algoritmo de roteamento Epidêmico é considerado a primeira proposta de roteamento para redes OppNet, segundo [79]. 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 3.6 apresenta o funcionamento do algoritmo Epidêmico.

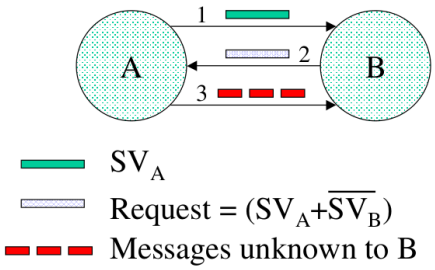


Figura 3.6: Funcionamento do algoritmo de roteamento Epidêmico quando dois nós A e B estão no mesmo raio de transmissão (fonte: [79])

.

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, os objetivos do algoritmo são maximizar a taxa de entrega, minimizar a latência e minimizar o total de recursos consumidos na rede.

- **PROPhET [36]** – o algoritmo de roteamento *Probabilistic Routing Protocol using History of Encounters and Transitivity - PROpHET*, foi proposto em 2003 por [36]. 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 superior. Dessa maneira esse algoritmo é um interessante *benchmark* utilizado na avaliação de novos protocolos para redes OppNet.

- **BUBBLE-RAP [20]** – o protocolo *Bubble Rap Forwarding* [20] é considerado o primeiro algoritmo de roteamento para OppNets baseado em características sociais. Basicamente, este algoritmo toma as decisões de roteamento usando os conceitos de comunidade e centralidade. Nesse algoritmo, quando uma oportunidade de contato ocorre, mensagens são encaminhadas primeiramente para aqueles

nós com alta centralidade e/ou membros da mesma comunidade a que pertence o destinatário da mensagem. Para realizar os cálculos necessários para detecção das comunidades e centralidades dos nós da rede, os autores utilizaram técnicas de grafos como k-clique e um algoritmo centralizado de pesos das arestas do grafo que representa a rede.

- **Friendship routing (FS) [6]** – esse é o primeiro algoritmo que considera a força da amizade entre os nós nas decisões de roteamento. Basicamente, ele encaminha mensagens para nós com uma amizade forte com o destinatário da mensagem. Para modelar a amizade entre nós, os autores consideram características ligadas diretamente aos contatos entre os nós, tais como sua frequência, sua longevidade e sua regularidade. Os autores afirmam que um par de nós são amigos se eles têm contatos regulares e de longa duração. Com base nessas premissas, uma nova métrica chamada *Social Pressure Metric* (SPM) é proposta para representar a pressão social que motiva os amigos a visitarem uns aos outros e compartilharem suas experiências.

É importante destacar que esses algoritmos não foram projetados considerando a existência de nós egoístas na rede. Desta forma, nas simulações realizadas, modificaram-se esses algoritmos para considerar o egoísmo entre os nós. Para alcançar este objetivo, cada um desses algoritmos tira proveito do sistema de avaliação de egoísmo descrito na Seção 3.3. Assim, se o nó *relay* selecionado por eles forem avaliados como egoístas, a mensagem não será enviada. Caso contrário, os algoritmos usam suas decisões originais de roteamento.

3.7 Métricas de interesse

Tomando como base outros trabalhos encontrados na literatura [20, 6, 36], utilizou-se na avaliação de desempenho realizada no presente trabalho as seguintes métricas:

- **Taxa de entrega** - mede a quantidade de número de mensagens recebidas em razão do número de mensagens enviadas.

- **Atraso médio de entrega** - mede o intervalo de tempo médio entre o envio e o recebimento das mensagens na rede.

- **Custo médio** - mede a quantidade média de eventos de encaminhamento por mensagem entregue ao destino.

3.8 Consumo de energia

Para simular o consumo de energia dos nós, utilizou-se o modelo de energia proposto em [62]. O consumo de energia de um nó é classificado em cinco estados:

- **Desativado** - não há consumo de energia, pois, a interface do nó da rede está desativada.
- **Inativo** - reduz o consumo de energia, pois, a interface do nó da rede está ociosa.
- **Escaneamento** - o nó consome energia enquanto a interface do nó da rede detecta vizinhos.
- **Transmissão** - o nó consome energia ao enviar uma mensagem.
- **Recepção** - o nó consome energia porque está recebendo uma mensagem.

Nas simulações realizadas, todos os nós inicialmente possuiam o nível máximo de energia. O nível de energia dos nós é medido em unidades, sendo 500 unidades o valor máximo. Considerou-se que o usuário recarrega seu dispositivo a cada 24 horas. O consumo de energia depende do estado do dispositivo e do número de operações usando a interface de rede. Por exemplo: se o nó está nos estados de transmissão, recepção ou escaneamento, assumiu-se uma redução de 25 unidades de energia para cada mensagem enviada/recebida e/ou para cada escaneamento realizado. Se o nó estiver nos estados inativo ou desativado, assume-se que não há custo de energia.

Capítulo 4

Roteamento de mensagens em OppNets

Em OppNets, a cada oportunidade de contato, o nó remetente tenta enviar suas mensagens para outros nós que tenham uma boa probabilidade de encontrar os destinatários das mensagens em contatos posteriores [39]. No entanto, como nestes tipos de rede o caminho desde um nó origem até um destino é intermitente, os algoritmos de roteamento convencionais geralmente não são aplicáveis [85]. Consequentemente, novos algoritmos de roteamento são necessários para superar os longos atrasos e problemas de frequentes desconexões [12] introduzidos nesse novo paradigma de comunicação. Em geral, o desafio tradicional de um algoritmo de roteamento em OppNets é construir um caminho entre um par de nós, permitindo a troca de mensagens entre eles. Uma boa maneira de atingir esse objetivo é selecionando os nós *relay* com boas chances de encontrar o destino em seus próximos contatos. É fácil perceber que quanto melhor for essa seleção, maiores serão as chances de entregar uma mensagem com sucesso.

Outro desafio-chave de um algoritmo de roteamento está relacionado à detecção de nós egoístas na rede. Em OppNets, um nó tem a autonomia de decidir se aceita ou não o recebimento de uma mensagem; consequentemente, a cooperação por parte de todos os nós não é totalmente garantida [23]. Dessa forma, se a rota construída pelo algoritmo de roteamento passa por um nó *relay* que é egoísta, este pode se recusar a cooperar caso o destinatário da mensagem seja um nó desconhecido ou com o qual ele simplesmente não deseja cooperar. Nesse caso, o que ocorre se o nó selecionado como *relay* não estiver disposto a receber ou carregar a mensagem? Além disso, de acordo com [31], os usuários no mundo real são egoístas e tendem a colaborar com um conjunto restrito de pessoas. Considerando essas suposições, é fácil perceber que um bom algoritmo de roteamento também precisa ser capaz de detectar se um nó é

egoísta ou não, e também descobrir o conjunto de nós com que cada nó está disposto a colaborar.

Neste Capítulo apresenta-se uma proposta de roteamento para OppNets, o algoritmo "*Friendship and Selfishness Forwarding*" (FSF), que toma suas decisões de roteamento considerando os relacionamentos sociais e o egoísmo dos nós. Primeiramente, para escolher nós *relay* para uma mensagem, FSF usa os laços sociais entre os nós como critério. A motivação para isso é que quanto mais forte for o laço social entre os nós, maior será a probabilidade deles se encontrarem no futuro. FSF divide os relacionamentos sociais entre os nós em três grupos: amigos, conhecidos e desconhecidos. Para descobrir os relacionamentos entre os nós, FSF usa as técnicas apresentadas no Capítulo 3. O próximo passo executado por FSF é a verificação do egoísmo do nó candidato a *relay*. Para realizar essa verificação, FSF usa o mecanismo apresentado na Seção 3.3. Finalmente, antes de tomar a decisão de roteamento FSF checa os níveis dos recursos dos nós. O objetivo é considerar os casos em que apesar da existência de relacionamento social com o destinatário da mensagem, o nó candidato a *relay* decide não receber uma mensagem porque os recursos do seu dispositivo estão em níveis críticos.

4.1 Trabalhos relacionados a Roteamento em OppNets

Na literatura para OppNets, vários algoritmos de roteamento estão disponíveis. Zhu et al. [91] classificam os algoritmos de roteamento de acordo com suas abordagens em três categorias: (i) baseados em nós adicionais, (ii) oportunísticos, e (iii) baseados em previsibilidade. Nos últimos anos, os algoritmos baseados em relações sociais surgiram como uma quarta categoria.

Os algoritmos de roteamento baseados em nós adicionais realizam a distribuição de dados em uma rede combinando o mecanismo de armazena-carrega-encaminha com o uso de nós adicionais, chamados de mulas de dados, que atuam como intermediários [91]. Muitos algoritmos de roteamento adotaram esse paradigma de encaminhamento, tais como [59, 87, 88, 89]. Mais recentemente, Burns et al. [9] propuseram um algoritmo de roteamento baseado nesse paradigma em que as decisões de roteamento são tomadas considerando encontros observados entre os pares de nós e as visitas dos nós às localizações geográficas. Os algoritmos de roteamento baseados em nós adicionais compartilham um problema comum: a sobrecarga e o custo extra para controlar os nós adicionais.

Nos algoritmos baseados no paradigma denominado oportunista, os nós trocam dados uns com os outros quando ocorre um contato, ou seja, sempre que estão dentro da mesma área de cobertura. O algoritmo de roteamento Epidêmico proposto em [79] é um exemplo de um algoritmo baseado nesse paradigma de encaminhamento. Este algoritmo de roteamento assume que, quando um par de nós está na mesma área de cobertura, eles enviam todas as mensagens armazenadas em seus *buffers* entre si, aumentando a probabilidade de entrega da mensagem.

Ainda se baseando nas oportunidades de contato, alguns pesquisadores exploram algumas características da rede para propor algoritmos de encaminhamento. Por exemplo, Shaghaghian e Coates [58] usam algumas suposições simplificadoras sobre o comportamento da rede para propor dois algoritmos de encaminhamento. O objetivo principal é reduzir as latências esperadas entre qualquer nó na rede para um destino específico em algumas situações. Os autores consideram que os algoritmos de encaminhamento para OppNets devem resultar em baixa latência média e em um uso eficiente dos recursos da rede. Com base nos resultados de suas simulações, os autores confirmam que os algoritmos propostos são capazes de melhorar tanto a latência quanto a taxa de entrega.

Da mesma forma, inspirado na solução cooperativa de Nash, Li et. al propõem em [29] o algoritmo de encaminhamento GameR para OppNets. Esse algoritmo usa uma função de utilidade derivada da taxa estimada de utilização de recursos e da previsibilidade de entrega baseando-se em dados históricos. De acordo com os resultados obtidos, os autores concluem que GameR pode usar eficientemente os recursos da rede, melhorando a taxa de entrega e diminuindo a sobrecarga nos nós em situações com recursos limitados. Outras propostas baseadas nesse paradigma incluem [74, 21, 8] e [75].

Algoritmos baseados em previsão são um refinamento da abordagem baseada em oportunidades de contato. Esses algoritmos tomam as suas decisões de roteamento baseando-se em algumas métricas. Geralmente, essas métricas são baseadas em características como a probabilidade de entrega de mensagens para o nó de destino [36, 38, 47], ou com base no histórico de encontros entre os nós da rede [8, 50, 86].

Recentemente, vários pesquisadores têm considerado o uso de algumas características sociais no projeto de novos protocolos de roteamento para OppNets. De acordo com Zhu et al. [91], os comportamentos que representam a utilização de dispositivos móveis, em geral, pode ser melhor descrito por modelos de redes sociais, uma vez que os dispositivos que compõem a rede são manipulados por seres humanos. Entre as características sociais que têm sido considerada no projeto de novos algoritmos para OppNets pode-se citar o altruísmo [48], a amizade [7, 54, 69], o egoísmo [31, 69], a

centralidade e a popularidade [20].

Hui et al. propõem em [20] o protocolo *Bubble Rap Forwarding*, considerado o primeiro algoritmo de roteamento para OppNets baseado em características sociais. Basicamente, este algoritmo toma as decisões de roteamento usando os conceitos de comunidade e centralidade. Nesse algoritmo, quando uma oportunidade de contato ocorre, mensagens são encaminhadas primeiramente para aqueles nós com alta centralidade e/ou membros da mesma comunidade a que pertence o destinatário da mensagem.

Em [6], Bulut et al. propõem o primeiro algoritmo de roteamento baseado na amizade entre os nós. Para modelar a amizade entre nós, os autores consideram características ligadas diretamente aos contatos entre os nós, tais como sua frequência, sua longevidade e sua regularidade. Os autores afirmam que um par de nós são amigos se eles têm contatos regulares, de longa duração e com uma determinada frequência. Com base nessas premissas, uma nova métrica chamada *Social Pressure Metric* (SPM) é proposta para representar a pressão social que motiva os amigos a visitarem uns aos outros e compartilharem suas experiências. O algoritmo proposto realiza o encaminhamento de mensagens se, e somente se, os nós envolvidos no contato tem uma amizade forte com os destinatários das mensagens.

Li et al. propõem um algoritmo de roteamento baseado no egoísmo dos nós em [31]. SSAR (*Social Selfishness Aware Routing*) é um protocolo que introduz considerações de egoísmo em cenários de OppNets. Esse algoritmo de roteamento tenta compensar a perda de desempenho causada pelo comportamento egoísta dos nós alocando recursos (*buffers* e largura de banda) com base na prioridade do pacote. Os autores desse artigo consideram que questões de egoísmo deveriam ser integradas em novos algoritmos de roteamento, já que as pessoas são socialmente egoístas, ou seja, estão dispostas a encaminhar mensagens para um número limitado de pessoas, e essa disposição depende da força do relacionamento entre elas. SSAR encaminha mensagens entre nós apenas se eles tiverem um relacionamento social forte. Por exemplo, se a relação social entre o nó A e o nó B for forte, o nó A sempre aceitará carregar consigo para encaminhar posteriormente mensagens para o nó B (e vice-versa).

Na estratégia seguida pelo algoritmo FSF, assim como os trabalhos citados acima, a modelagem da amizade entre os nós considera algumas informações sobre o histórico de contatos entre os nós, tais como a duração, frequência e tempo entre os contatos. Além disso, FSF também considera o número de encontros fora da universidade/trabalho, pois, se acredita que essa informação seja uma boa evidência da existência de amizade entre dois nós. No mundo real, é razoável supor que quanto mais encontros fora do trabalho e/ou universidade existir entre um par de nós, mais forte

será o relacionamento social entre eles. O algoritmo de roteamento FSF também considera o egoísmo do candidato a *relay* da mensagem nas decisões de encaminhamento. Diferentemente dos trabalhos existentes na literatura, FSF considera que em diversas situações um nó A pode não aceitar as mensagens cujo nó de destino é o nó B porque A deseja economizar os recursos do seu dispositivo, independentemente de quão forte seja a sua amizade com o nó B. Finalmente, FSF também considera duas categorias de comportamento egoísta: aqueles em que o nó é egoísta apenas sob condições específicas (por exemplo, restrições de recursos), e aquele em que o nó se comporta de maneira egoísta em qualquer situação.

4.2 Algoritmo FSF: estratégia de encaminhamento

Baseando-se nas premissas citadas anteriormente, nesta Seção, apresenta-se a estratégia de encaminhamento seguida pelo algoritmo FSF. Quando surge uma oportunidade de contato, FSF encaminhará uma mensagem se, e somente se, os nós forem amigos ou conhecidos (relação social forte), se o nó *relay* não for egoísta ou socialmente egoísta e seu dispositivo não tiver restrições de recursos. Caso contrário, a mensagem não será encaminhada. Por exemplo, se um nó A tendo uma mensagem M1 endereçada ao nó B encontra com o nó C, o nó A encaminhará M1 para o nó C se, e somente se, as seguintes suposições forem verdadeiras: os nós B e C são amigos ou conhecidos um do outro, o nó C é não egoísta ou socialmente egoísta, e os recursos do dispositivo do nó C não estão em níveis críticos. Se o nó C for socialmente egoísta, ele receberá a mensagem M1 somente se B pertencer ao seu círculo de amigos. Caso contrário, a mensagem não será encaminhada. Além disso, se o nó A encaminha a mensagem para o nó C, a mensagem M1 não será excluída do *buffer* do nó A para aumentar a probabilidade de entrega da mensagem. Finalmente, se os nós B e C forem desconhecidos um do outro, ou se o nó C for avaliado como egoísta individualmente, ou ainda se os recursos do dispositivo do nó C estiverem em níveis críticos, a mensagem não será encaminhada. A Figura 4.1 mostra o fluxograma representando a estratégia de encaminhamento da FSF.

4.3 Modificações no *The One Simulator*

Para avaliar o algoritmo FSF, realizaram-se simulações dirigidas por traces usando o simulador *The One Simulator* [22]. Inicialmente, incluiu-se no simulador a implementação do algoritmo FSF. Ao todo, incluiu-se duas novas classes no simulador, intituladas *FriendshipClassifier* e *SelfishnessAssessment*. Essas duas classes implementam os me-

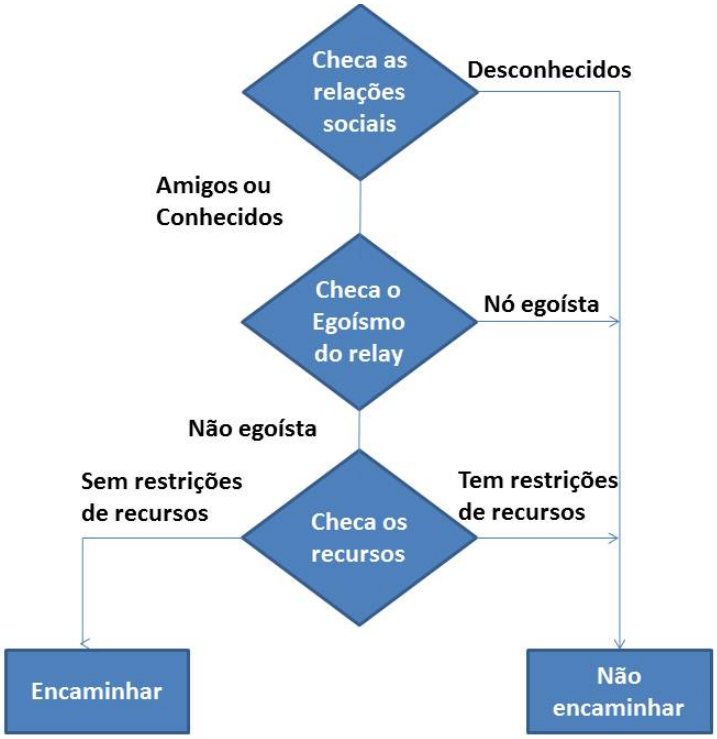


Figura 4.1: Fluxo da estratégia de encaminhamento seguida pelo algoritmo FSF.

canismos mostrados nas Seções 3.3 e 3.4, respectivamente. Para usar o mecanismo de aprendizado de máquina descrito na Seção 3.4.2, integrou-se o simulador The One com a ferramenta WEKA [83]. WEKA é uma ferramenta implementada em Java, que contém as implementações de vários algoritmos de aprendizado de máquina. Também foram realizadas modificações nas classes *MessageRouter*, *ActiveRouter*, *DTNHost*, e *DTNSim*. Os algoritmos usados para comparação, os geradores de eventos (leitores de mobilidade e geração de mensagens) e os relatórios usados para análise são aqueles disponíveis no pacote *The One Simulator*.

4.4 Resultados e discussões

Nas simulações realizadas na presente avaliação de desempenho, comparou-se o desempenho de FSF com o dos algoritmos citados na seção 3.6 considerando os quatro cenários de mobilidade citados em 3.5 . Foram avaliados dois cenários diferentes:

- cenário com variação do TTL da mensagem – neste cenário, os nós têm tamanho de *buffer* ilimitado, ou seja, eles podem armazenar suas próprias mensagens, assim como as mensagens endereçadas a outros nós com os quais eles aceitam

colaborar. Quando o TTL expira, FSF descarta automaticamente do *buffer* a mensagem;

- cenário com variação do tamanho do *buffer* – neste cenário, os nós têm TTL ilimitado, ou seja, não há limite de tempo para armazenar uma mensagem, mas os *buffers* têm espaço de armazenamento secundário limitado.

As métricas de interesse são as citadas na Seção 3.7. Derivado de trabalhos similares encontrados na literatura, o tamanho da mensagem varia de 512 KB a 1 MB, enquanto o tamanho do *buffer* varia de 10 MB a 60 MB, e os valores de TTL variam de 5 a 96 horas. Nas simulações utilizou-se a distribuição uniforme para gerar 1000 mensagens entre pares de nós. Em cada experimento realizado construiu-se para cada resultado obtido um intervalo de confiança de 95% para os parâmetros estimados. Em ambos os cenários utilizou-se o algoritmo *Drop Oldest* [70] como a estratégia de gerenciamento de *buffer*.

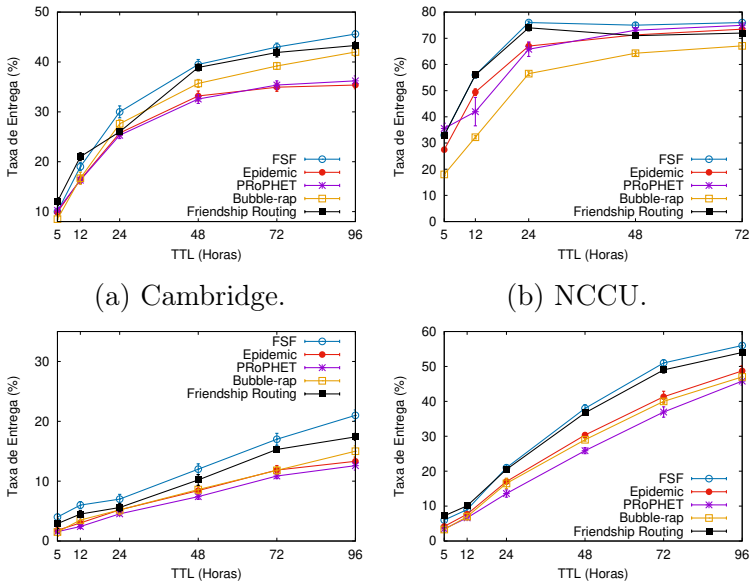
4.4.1 Resultados variando o tamanho do TTL

As Figuras 4.2, 4.3 e 4.4 apresentam os resultados da variação dos valores do TTL da mensagem. A expectativa é de que o uso de relações sociais entre os nós como critério de encaminhamento aumente a taxa de entrega na rede por conta dos encontros que naturalmente ocorrem entre os nós que mantém uma relação social. Os resultados mostrados na Figura 4.2 confirmam essa intuição, uma vez que o algoritmo FSF alcançou a melhor taxa de entrega em todos os cenários, entregando até 7% mais mensagens do que os outros algoritmos. Vale ressaltar que em todos os cenários testados os algoritmos se comportam de maneira bastante semelhante, ou seja, quanto maior o TTL da mensagem, maior é a quantidade de mensagens entregues. Além disso, o algoritmo FS obtém a segunda melhor taxa de entrega em todos os cenários testados, confirmando que o uso de laços sociais (amizade, conhecidos, vizinhos, etc.) é um interessante critério que pode contribuir para o aumento da probabilidade de entrega de mensagens em OppNets.

Com relação ao atraso médio de entrega e ao custo médio, os algoritmos exibem o mesmo comportamento: quanto maior o TTL da mensagem, maior é o atraso na entrega e maior é o custo médio. Além disso, para todos os algoritmos, quanto maior a taxa de entrega, maior o atraso na entrega. Como o algoritmo FSF entregou mais mensagens, também obtém o maior atraso na entrega. Em relação ao custo médio, FSF alcançou o melhor resultado em Cambridge e NCCU. O algoritmo Bubble-Rap

obtém o melhor resultado em Sassy, enquanto no trace *Reality* o algoritmo PRoPHET teve um melhor desempenho.

Certamente, a quantidade de relacionamentos detectados entre os nós, bem como o tempo entre contados entre os nós que tem uma relação social, impacta o desempenho do algoritmo FSF. Quanto maior o número médio de relações sociais por nó, maior a probabilidade dele encaminhar uma mensagem durante uma oportunidade de contato. Esse fato também pode contribuir para o aumento do custo médio de encaminhamento do algoritmo FSF. Por outro lado, quando o tempo entre contatos dos nós com o relacionamento social é alto, consequentemente ocorrerá um aumento no atraso médio de entrega. A partir dos resultados das simulações, é razoável supor que o número médio de relações sociais detectadas por FSF para cada nó é baixo e o tempo entre contatos é alto, fatos que contribuem para aumentar o atraso médio do FSF e reduzir o número de encaminhamento de mensagens.



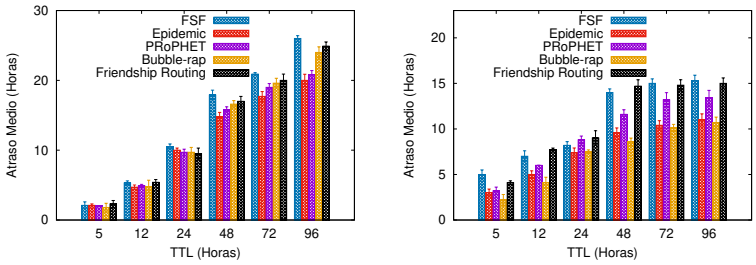
(a) Cambridge. (b) NCCU.

(c) Sassy. (d) Reality.

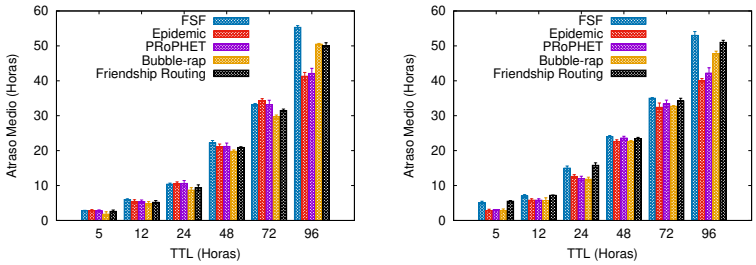
Figura 4.2: Taxa de entrega ao variar o tamanho do TTL da mensagem.

4.4.2 Resultados da variação do tamanho do *buffer*

As Figuras 4.5, 4.6 e 4.7 apresentam os resultados da variação dos valores do tamanho do *buffer* dos nós. Com relação à taxa de entrega, em todos os cenários testados, os algoritmos se comportam de forma bastante semelhante, ou seja, quanto maior o tamanho do *buffer*, maior é a taxa de entrega. O algoritmo FSF superou os outros algoritmos nos cenários de Cambridge e Sassy, alcançando taxas de entrega até 18%

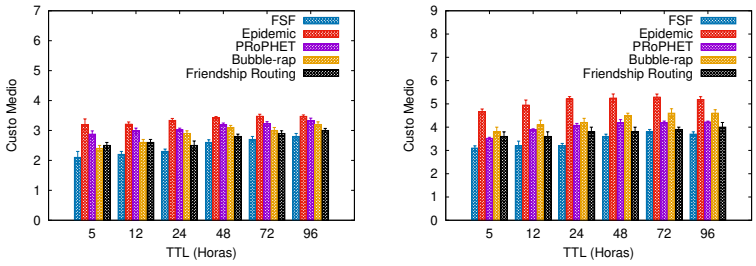


(a) Cambridge. (b) NCCU.

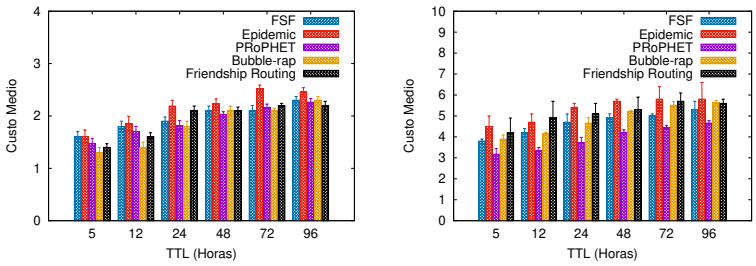


(c) Sassy. (d) Reality.

Figura 4.3: Atraso médio de entrega ao variar o tamanho do TTL da mensagem.



(a) Cambridge. (b) NCCU.



(c) Sassy. (d) Reality.

Figura 4.4: Custo médio de encaminhamentos ao variar o tamanho do TTL da mensagem.

e 5% maiores que os demais algoritmos, respectivamente. O algoritmo FS, também baseado em características sociais, obtém a segunda melhor taxa de entrega em todos os cenários. A estratégia de roteamento PRoPHET alcançou a melhor taxa de entrega no cenário NCCU (até 20% maior), seguida pelo roteamento Epidêmico.

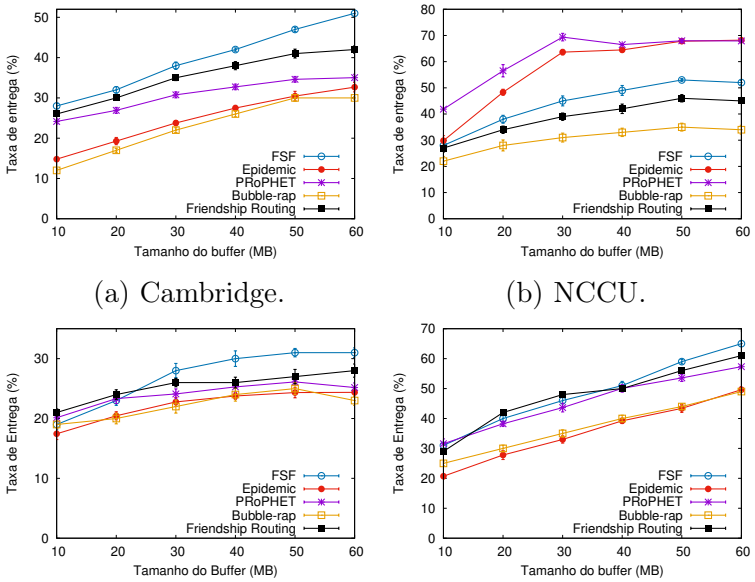
Novamente, quanto maior a taxa de entrega, maior é o atraso médio de entrega também. Por exemplo, nos cenários de Cambridge e Sassy o algoritmo FSF obteve as maiores taxas de entrega e atraso médio de entrega. A mesma regra se aplica ao algoritmo PRoPHET no cenário NCCU e FS no cenário *Reality*. Com relação ao custo médio, o algoritmo FSF também superou os demais algoritmos em todos os cenários testados. Além disso, vale a pena enfatizar que FSF também alcançou a maior taxa de entrega nos cenários de Cambridge, Sassy e *Reality*.

Em relação ao custo médio de encaminhamentos, nota-se a maior diferença entre os resultados alcançados pelos algoritmos testados. De maneira geral, FSF alcançou o menor custo médio em todos os cenários testados. No entanto, destacam-se os resultados alcançados por esse algoritmo nos cenários Cambridge, Sassy e *Reality*, nos quais, FSF também alcançou a melhor taxa de entrega. Pode-se justificar esses resultados através das rotas construídas por FSF. Esses resultados mostram que as rotas construídas por FSF podem conectar um par de nós usando menos encaminhamentos. Baseando-se nesses resultados, pode-se concluir que FSF contribui para o aumento da probabilidade de entrega de mensagens e diminui o custo médio de encaminhamentos, uma clara evidência de economia dos recursos dos nós.

É importante também realizar uma comparação dos resultados obtidos em ambas as variações realizadas. De maneira geral, os algoritmos testados tiveram melhor desempenho ao variar o TTL da mensagem do que o tamanho do *buffer*. Nos cenários com variação do TTL, a taxa de entrega foi até 10% melhor (Sassy e *Reality*) do que quando se varia o tamanho do *buffer*. Destacam-se também os resultados do atraso médio de entrega, para o qual nos cenários Sassy e *Reality* a diferença entre o maior atraso de entrega é de até 110 horas (Sassy). Pode-se concluir que o aumento observado na taxa de entrega também contribuiu para um aumento da taxa de entrega. Da mesma forma, para o custo médio de encaminhamentos também percebe-se diferenças maiores. Por exemplo, no cenário Cambridge essa diferença era de 15 encaminhamentos ao variar o tamanho do *buffer*. Essa diferença é de até 30 encaminhamentos no cenário *Reality*. A intuição por trás desses resultados é muito simples: quanto maior o TTL da mensagem, mais tempo um nó tem para entregá-la com sucesso ao destino ou para encaminhá-lo para outro nó *relay*. Isso justifica o maior atraso de entrega e o custo médio alcançado por todos os algoritmos testados nos cenários em que o valor de TTL é infinito. Por outro lado, quando o TTL da mensagem é limitado, o tempo entre contatos entre os nós

desempenha um papel importante. Quanto maior o tempo entre contatos dos nós na rede, maior o número de mensagens descartadas do *buffer* pela expiração do TTL. Isso justifica o menor custo médio alcançado por todos os algoritmos testados no cenário com tamanho de *buffer* infinito, ou seja, que também tem TTL finito.

Finalmente, destaca-se o impacto de nós egoístas na rede, seja porque o dispositivo do nó está em restrições de recursos, ou seja, porque ele é racionalmente egoísta. Os resultados obtidos confirmam que a existência de nós egoístas degrada o desempenho da rede, assim como é mostrado em outros trabalhos encontrados na literatura [60, 41, 30]. De maneira geral, a partir dos resultados, pode-se perceber esse impacto principalmente na taxa de entrega, para qual o melhor índice alcançado é próximo de 70% de mensagens entregues no cenário NCCU. Acredita-se que isso se deva à maior quantidade de contatos observada nesse cenário (conforme mostrado na Tabela 3.2).



(a) Cambridge. (b) NCCU.

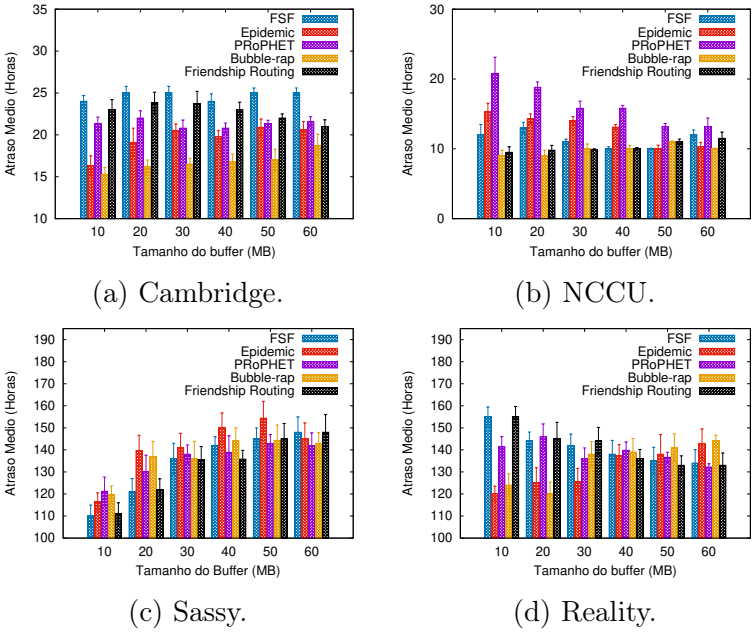
(c) Sassy. (d) Reality.

Figura 4.5: Taxa de entrega ao variar o tamanho do buffer dos nós.

4.4.3 Análise de complexidade de FSF

Uma importante questão a ser considerada é a complexidade dos procedimentos necessários para o funcionamento do algoritmo FSF. Essa análise pode ser dividida em duas partes;

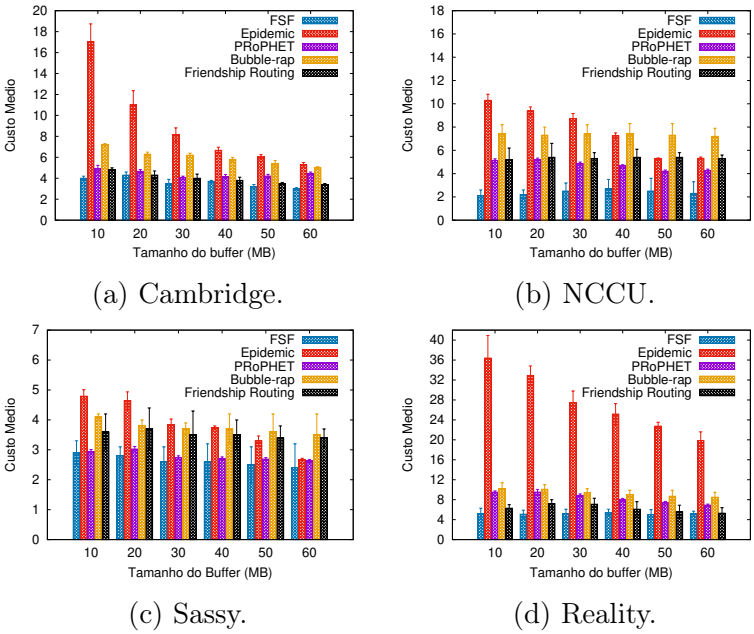
- **Tempo de transmissão das estruturas e tamanho ocupado na memória** – para tomar suas decisões, o sistema SCRS utiliza duas estruturas de dados: uma



(a) Cambridge. (b) NCCU.

(c) Sassy. (d) Reality.

Figura 4.6: Atraso médio de entrega ao variar o tamanho do buffer dos nós.



(a) Cambridge. (b) NCCU.

(c) Sassy. (d) Reality.

Figura 4.7: Custo médio de encaminhamentos ao variar o tamanho do buffer dos nós.

lista e uma matriz. O tamanho de cada estrutura é diretamente proporcional ao número de nós com quem cada nó interage. Nessa análise de complexidade, considere os seguintes pontos:

- Cada posição da lista tem 2 atributos: um ID e um valor que representa a relação social (amigos, conhecidos, desconhecidos). Ambos são representados por inteiros.

- Cada posição da matriz tem apenas um atributo: um valor que representa a relação social, também representado por um inteiro.

- O simulador utilizado para simulação é escrito em Java. Nessa linguagem, um inteiro ocupa 4 bytes.

Considere que um nó interage com outros 100 nós. Nesse cenário, a lista terá 100 elementos e a matriz global terá 100 linhas $\times$ 100 colunas. Desta forma, o tamanho da lista é:

Tamanho da lista: 100 elementos $\times$ 4 bytes $\times$ 4 bytes = 160 bytes ou 0,16 KB

Tamanho da Matriz: 100 $\times$ 100 $\times$ 4 bytes = 40.000 bytes ou 40 KB.

Considerando um link *Bluetooth* de 2 Mbps, por exemplo, são necessários de 0,16 segundos para o completo envio da matriz de um nó para outro. Este valor é considerado um tempo razoável.

Com relação a este ponto, pode-se considerar dois fatos:

- A partir dos resultados apresentados no trabalho, pode-se observar que FSF diminui o número de encaminhamentos. Desta forma, pode contribuir para diminuir o impacto causado pelo envio da matriz utilizada pela FSF.

- Na maioria das vezes cada nó não interage com mais de 100 outros nós. Dessa forma, é razoável supor que o tamanho da matriz utilizada é menor que 40Kb. Consequentemente, o tempo de transmissão da matriz entre dois nós é ainda menor que 0,16 segundos.

- **Complexidade dos algoritmos utilizados** – com relação à complexidade, em ambos os casos, uma vez que as informações sobre os IDs dos nós estão disponíveis, a complexidade para acessar as informações é:

  Lista: O(N) no pior dos casos, porque o algoritmo utilizado precisa encontrar o elemento na lista, ou seja, no pior dos casos buscará em todos os N elementos que compõem a lista. Matriz: O(1), porque os índices estão disponíveis e o algoritmo utilizado não precisa procurar nenhum item, apenas acessá-lo diretamente e alterar seu valor.

4.4.4 Discussões Adicionais

Uma questão importante que afeta o desempenho do algoritmo FSF é a classificação correta da amizade entre nós na rede. Neste trabalho, introduz-se uma abordagem para detectar a existência da amizade entre os nós usando um algoritmo de aprendizado de máquina. Esse algoritmo tem duas tarefas importantes: primeiro, aprender sobre amizade de nós com base em dados coletados do mundo real. A segunda tarefa é classificar novos relacionamentos entre dois nós na rede. A partir dos resultados, é razoável supor que Naive Bayes ofereça um desempenho promissor para classificar a amizade entre os nós. Dessa forma, quanto melhor a classificação realizada pelo algoritmo Naive Bayes, melhor será o desempenho do algoritmo FSF.

A implementação de um algoritmo de aprendizado de máquina mostrou ser uma solução interessante para a tarefa de detectar a existência da amizade entre dois nós. No entanto, algumas questões devem ser consideradas. Por exemplo, a disponibilidade de informações sobre o que é amizade no mundo real pode ser um requisito importante, pois, em alguns cenários, essas informações não estão disponíveis. Para resolver este problema, pode-se monitorar o cenário através de uma aplicação que seja responsável por coletar informações sobre a amizade entre os nós. Outra alternativa é usar um algoritmo de aprendizado de máquina que não precise de um banco de dados de treinamento para classificar novas instâncias.

Finalmente, considera-se que o uso de uma abordagem de aprendizado de máquina pode ser mais flexível do que outras abordagens para classificar a amizade entre nós. Por exemplo, se houver a necessidade de alterar as características utilizadas no modelo de amizade, usar o aprendizado de máquina pode acelerar esse processo. Outra vantagem é que uma abordagem de aprendizado de máquina pode combinar várias características para classificar a amizade e, assim, pode facilmente usar dados de situações do mundo real.

4.5 Considerações finais

Neste Capítulo, apresentou-se uma proposta de roteamento que inclui o uso dos relacionamentos sociais como critério para a disseminação de mensagens em OppNets. Dessa forma, apresenta-se o algoritmo de roteamento *Friendship and Selfishness Forwarding* (FSF) que considera duas informações nas decisões de roteamento: os relacionamentos sociais entre os nós e o egoísmo dos nós candidatos a *relay*. FSF realiza duas tarefas principais: em primeiro lugar, FSF classifica os laços sociais entre os nós; segundo, FSF usa um sistema de reputação para verificar o egoísmo do nó *relay* considerando os

casos em que, apesar de uma forte relação social com o destino, o nó *relay* pode recusar o recebimento da mensagem porque ele é egoísta ou seu dispositivo possui restrições de recursos naquele instante.

Para validar o algoritmo proposto, realizou-se um conjunto de experimentos para determinar a eficácia da entrega de mensagens em quatro cenários baseados em simulações controladas por traces usando o simulador *The ONE*. Em comparação com outros algoritmos de roteamento da mesma classe, FSF ofereceu melhores resultados em relação a um conjunto de métricas padrão, ou seja, a taxa de entrega, o custo médio, além de alcançar resultados razoáveis de atraso de entrega. Baseando-se nesses resultados, conclui-se que FSF contribui para o aumento da probabilidade de entrega de mensagens e diminui o custo médio de encaminhamentos, uma clara evidência de economia dos recursos dos nós.

Capítulo 5

Gerenciamento de buffer em OppNets

Os dispositivos móveis que compõem as OppNets compartilham de outro problema em comum: a limitação de recursos. A memória secundária dos dispositivos, doravante referida como espaço em *buffer* é um destes recursos escassos e limitados. Segundo [34], para incrementar a probabilidade de entrega, o roteamento em OppNets pode requerer que várias cópias de uma mesma mensagem seja replicada na rede. Dessa forma, combinando-se essas estratégias de roteamento 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 OppNets, esse fenômeno é chamado de *buffer overflow*. Quando ocorre o *overflow*, uma ou mais mensagens devem ser selecionadas para serem retiradas do *buffer*, permitindo-se assim armazenar-se a nova mensagem. O problema do gerenciamento de *buffer* em OppNets pode então ser definido como a seleção das mensagens que serão retiradas do *buffer* em casos de *overflow*. Essa escolha é de suma importância para maximizar o desempenho da rede, pois, ela influencia diretamente o roteamento de mensagens e, consequentemente, a entrega de mensagens na rede.

5.1 Trabalhos relacionados ao gerenciamento de *buffer* em OppNets

De acordo com [2], os algoritmos de gerenciamento de *buffer* em OppNets podem ser divididos em três tipos: (i) Algoritmos baseados em dados globais da rede, (ii) abordagens baseadas em dados locais e, (iii) algoritmos tradicionais. Da mesma forma que

para os algoritmos de roteamento, recentemente uma nova categoria de algoritmos de gerenciamento de *buffer* surgiu, a dos algoritmos baseados em características sociais.

Em [57], Settawatcharawanit et al. propõem o primeiro algoritmo de gerenciamento de *buffer* baseado em características sociais encontradas na literatura para OppNets. Para realizar as decisões de descarte, o algoritmo proposto por esses autores considera uma estratégia baseada na centralidade e na comunidade dos nós da rede. Para detectar a comunidade dos nós, os autores usam um algoritmo de detecção de comunidade chamado k-clique. Para indicar a centralidade dos nós, os autores usam um algoritmo de aproximação de centralidade, chamado *C-Window*. Quando o *buffer* de um nó *relay* está cheio, o algoritmo de gerenciamento de *buffer* proposto por eles descarta primeiramente mensagens cujos destinatários são de diferentes comunidades do nó que toma a decisão. Se existem muitas mensagens para destinatários de uma mesma comunidade, o nó descarta a mensagem destinada ao nó com a menor centralidade.

Em [68], Souza et al. propõem um algoritmo denominado *Drop Less Known* (DLK) para o gerenciamento de *buffer* dos nós de redes DTN. Esse algoritmo introduz uma métrica denominada força da relação social, a qual considera o quão forte são os laços sociais entre dois nós (forte, médio, fraco). Para classificar a relação social entre um par de nós, os autores utilizaram a quantidade de contatos e um *log* de ligações entre os nós disponibilizados em um conjunto de dados denominado *Reality*. Os experimentos realizados mostraram que o algoritmo DLK contribuiu para um aumento na taxa de entrega e houve diminuição razoável do atraso médio de entrega de mensagens na rede.

Em [73] um algoritmo denominado *Selfish Drop-Based* (SDB) é proposto. Esse algoritmo considera o egoísmo dos nós que compõem a rede. Para isso, os autores utilizam a classificação do egoísmo dos nós realizada por [84], a qual divide o egoísmo de um nó em social ou individual. O nó egoísta individual é aquele que não aceita mensagens destinadas a outros nós; está interessado somente na sua participação dentro da rede. O nó egoísta social aceita mensagens destinadas a outros nós, porém, somente àqueles com quem ele tem alguma relação social. Basicamente, SDB realiza o gerenciamento de acordo com o tipo do nó: nó classificado como individual, descarta a mensagem candidata a entrar no *buffer*, nó classificado como social descarta primeiramente mensagens para nós classificados como não amigos. Outra contribuição interessante desse algoritmo proposto é que dependendo das limitações de seus recursos (nível de bateria, por exemplo), o nó pode tornar-se egoísta. Nó não egoísta e sem limitações de recursos, executa o descarte considerando a força da amizade entre os nós.

A Tabela 5.1 destaca outros trabalhos encontrados na literatura para OppNets nos últimos 5 anos.

5.2 Algoritmo Friendly-Drop

Baseando-se nas lacunas existentes na literatura, no presente trabalho apresenta-se uma nova proposta de algoritmo para gerenciamento do *buffer* dos nós que compõem a rede. Denominou-se esse algoritmo como *Friendly-Drop Algorithm* (FDA). O Algoritmo 1 apresenta o funcionamento do algoritmo FDA. A princípio, FDA classifica a prioridade das mensagens de acordo com a relação social entre os nós. Da mesma maneira que FSF, FDA também considera três tipos de relacionamentos sociais: amigos, conhecidos ou desconhecidos.

Quando surge uma oportunidade de contato, o nó encaminha as mensagens de acordo com o nível de prioridade. Primeiramente, o nó encaminha as mensagens endereçadas aos amigos do candidato a *relay*, seguidas por aquelas endereçadas ao conjunto de nós conhecidos pelo *relay* e, por fim, aquelas para destinatários desconhecidos.

Quando ocorre um *buffer overflow*, as prioridades são invertidas, ou seja, FDA primeiro retira as mensagens endereçadas a nós desconhecidos, seguidas por aquelas destinadas a nós do conjunto de conhecidos do nó e, por último, aquelas destinadas aos amigos do nó. Se todos os destinatários das mensagens tiverem a mesma prioridade (por exemplo, todos as mensagens armazenadas no *buffer* são destinadas a amigos ou desconhecidos), o nó que toma a decisão primeiramente descarta as mensagens mais antigas e encaminha primeiramente as mensagens mais novas no *buffer*.

5.3 Avaliação de desempenho do algoritmo Friendly-Drop

Para avaliar o desempenho do algoritmo FDA, utilizou-se a técnica da simulação. Especificamente, implementou-se um modelo de simulação que inclui o algoritmo FDA proposto, que foi implementado no simulador The ONE [69]. Nos cenários simulados, utilizaram-se as seguintes configurações:

- Mobilidade de nós - para simular a mobilidade de nós, utilizou-se dois traces de mobilidade real bem conhecidos na literatura, denominados Sassy e *Reality* [52, 67]. Maiores detalhes sobre esses traces são apresentados na Seção 3.5.

- Algoritmos de roteamento - para avaliar o impacto de diferentes algoritmos de gerenciamento de *buffer* no processo de entrega de mensagens, utilizou-se dois algoritmos de roteamento bem conhecidos, (i) Algoritmo Epidêmico [6] (porque é uma estratégia de replicação) e , (ii) FSF [69], (porque é uma estratégia de

Algorithm 1 Friendly-Drop Algorithm - divide o *buffer* do nó (M) nos seguintes grupos: (F) mensagens destinadas a amigos, (A) mensagens destinadas a conhecidos, e (D) mensagens para nós desconhecidos.

```
1: procedure FRIENDLYDROP(M, HE, L, F, A, D)
       ▷ HE é a História dos encontros e L contém a amizade auto-relatada entre os nós.
2:     for cada no da rede n_i ∈ N do
           n_i.FS ← discoverFS(n_i, L)   ▷ Onde discoverFS é o procedimento mostrado na seção 3.4.2
           n_i.ACS ← discoverACS(n_i, HE)   ▷ Onde discoverACS é o procedimento mostrado na Seção 3.4.1
3:     end for
4:     for cada mensagem m_i ∈ M do
5:         if m_i.dest ∈ n_i.FS then
6:             Inserir (m_i, F)   ▷ Grupo F de amigos do nó
7:         else
8:             if m_i.dest ∈ n_i.ACS then
                   ▷ Grupo A de conhecidos do nó
9:                 Inserir(m_i, A)
10:            else
11:                Inserir (m_i, D)   ▷ Destinatários desconhecidos
12:            end if
13:        end if
14:    end for
15:    while no A esta conectado ao no B do
16:        Encaminhar primeiro todas as mensagens na fila F
17:    end while
18:    if M esta cheio then
19:        while A.M esta cheio do
20:            Descarta primeiro mensagens da fila D
21:        end while
22:    end if
23: end procedure
```

encaminhamento e também baseada em características sociais). A Tabela 5.2 apresenta os valores usados como configurações de rede.

- Geração de mensagens - para cada simulação realizada foram criadas 1000 mensagens, geradas uniformemente entre todos os pares de nós. Além disso, para todos os resultados obtidos calcularam-se o 95-percentil usando a distribuição t-student.

Para fins de comparação, utilizou-se na literatura quatro algoritmos bem conhe-

cidos:

- *Evict most favorably forwarded first* (MOPR) – esse algoritmo foi proposto em [37]. Cada nó da rede mantém um valor de probabilidade para cada mensagem armazenada na sua fila. Essa probabilidade representa a probabilidade do nó destinatário ter recebido a mensagem. Cada vez que a mensagem é encaminhada, o valor da probabilidade é atualizado. Mensagens com alto valor de probabilidade de já terem alcançado os seus destinatários são descartadas primeiramente.

- *Drop Less Known* (DLK) [68] – esse algoritmo introduz uma métrica denominada força da relação social, a qual considera o quão forte são os laços sociais entre dois nós (forte, médio, fraco). Para classificar a relação social entre um par de nós, os autores utilizaram a quantidade de contatos e um *log* de ligações entre os nós disponibilizados em um conjunto de dados denominado *Reality*. Os experimentos realizados mostraram que o algoritmo DLK contribuiu para um aumento na taxa de entrega e houve diminuição razoável do atraso médio de entrega de mensagens na rede.

- *Last Recently Forwarded* (LRF) – esse algoritmo foi proposto em [45]. A ideia básica do algoritmo é que mensagens que não foram enviadas em um certo tempo devem ser descartadas primeiro. A intuição por trás dessa ideia é que se essas mensagens não foram encaminhadas há um certo tempo, existe uma probabilidade das mesmas já terem alcançado seus destinatários.

- *Drop Random* (Random) – é um dos algoritmos mais básicos de gerenciamento de *buffer* para OppNets. A mensagem que será descartada ou enviada é escolhida de maneira aleatória.

5.4 Resultados dos experimentos

Nesta Seção, apresentam-se os resultados obtidos para os cenários avaliados. O objetivo principal é comparar o desempenho do algoritmo de gerenciamento de *buffer* FDA com relação aos algoritmos MOPR, LRF, DLK e Random. Baseando-se em outras configurações obtidas através de outros trabalhos encontrados na literatura, avaliou-se o impacto da variação do tamanho do *buffer* de 10 a 80 MB em cada cenário. As métricas de interesse são aquelas citadas na Seção 3.7.

5.4.1 Resultados com roteamento Epidêmico

As Figuras 5.1 e 5.2 apresentam os resultados dos cenários Sassy e *Reality* testados usando o algoritmo de roteamento Epidêmico. Observe que, quando surge uma oportunidade de contato, os nós que usam o Epidêmico como um protocolo de roteamento enviam todas as mensagens armazenadas em seu *buffer* para o outro nó em contato. Apesar de aumentar a probabilidade de entrega das mensagens, essa característica do algoritmo também aumenta o grau de congestionamento da rede. Portanto, é razoável supor que o algoritmo de gerenciamento de *buffer* se torna muito importante para reduzir o impacto causado pelo congestionamento da rede, mantendo no *buffer* as mensagens que os nós tem mais probabilidade de entregar ao destinatário.

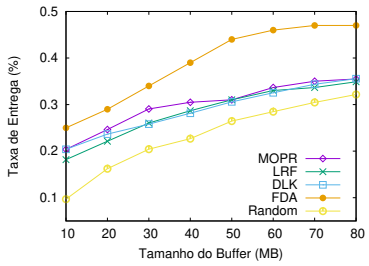
A partir dos resultados pode-se notar que em ambos os cenários, o uso de FDA como algoritmo de gerenciamento de *buffer* obteve os melhores resultados em termos de taxa de entrega de mensagens. Isso confirma a intuição levantada anteriormente: o uso dos relacionamentos sociais entre os nós contribuem para melhorar a taxa de entrega na rede. A intuição por trás disso é que, uma vez que pessoas que mantém alguma relação social tendem a se encontrar com uma dada frequência, e isso aumenta a probabilidade de entrega de uma mensagem entre eles. Da mesma forma, as informações dos relacionamentos sociais entre os nós pode contribuir para melhorar as decisões de gerenciamento de *buffer*. Portanto, é razoável supor que essas características são critérios interessantes para o gerenciamento de *buffer* do que outros testados.

Destaca-se também o resultado alcançado por FDA para a métrica média de encaminhamentos. A partir dos resultados apresentados nas Figuras 5.1.(c) e 5.2.(c), nota-se que FDA obteve os melhores resultados em ambos os cenários com relação a esta métrica. A diferença entre FDA e seus oponentes alcançam até cinco eventos de encaminhamentos a menos no cenário *Reality* e três eventos de encaminhamentos no cenário Sassy. Isso mostra claramente que FDA contribui para diminuir o número de encaminhamentos, consequentemente economizando recursos de rede. De fato, no mundo real, um par de amigos geralmente também compartilham de outros amigos em comum e, às vezes, até mesmo conhecidos, ou seja, têm vizinhos em comum. Desta forma, é razoável supor que, usando os nós amigos ou conhecidos de um nó, uma mensagem pode chegar ao seu destino mais rapidamente, e isso claramente diminui o número de encaminhamentos.

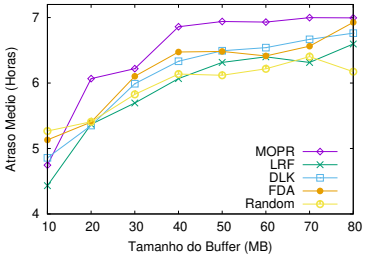
Figuras 5.1.(b) e 5.2.(b) apresentam os resultados com relação à métrica atraso médio de entrega. O algoritmo FDA obteve um resultado razoável no cenário Sassy, mas foi superado pelos outros algoritmos testados no cenário *Reality*. A principal razão para o maior atraso médio do algoritmo FDA está ligado à quantidade de mensagens

entregues pelo algoritmo de roteamento combinado com FDA. Quanto maior essa taxa de entrega, maior será o atraso médio.

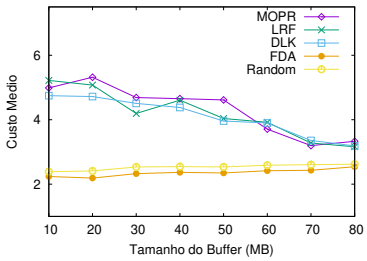
A intuição por trás desse resultado é que as mensagens adicionais que FDA entregou têm um atraso de entrega maior, visto que essas mensagens são mantidas no *buffer* do nó por um longo período e, consequentemente, o atraso médio aumenta. Apesar de os relacionamentos sociais poderem ser considerados um bom critério para melhorar a taxa de entrega, o seu uso pode aumentar o atraso de entrega de acordo com o tempo entre contatos de um par de amigos ou conhecidos. Quanto maior o tempo entre contatos, maior será o atraso médio de entrega. Desta forma, é razoável supor que nos cenários avaliados, o tempo entre contatos é alto. Uma abordagem interessante para diminuir o impacto causado pelo tempo entre contatos poderia ser a utilização de uma heurística baseada no TTL da mensagem e no tempo entre contatos. Se o TTL restante for maior que o tempo estimado entre contatos, FDA poderia descartar essa mensagem primeiramente. Por outro lado, baseado nessa mesma informação, FDA poderia automaticamente estender o TTL da mensagem para que a mesma se mantenha disponível para ser entregue em contatos posteriores.



(a) Taxa de entrega.

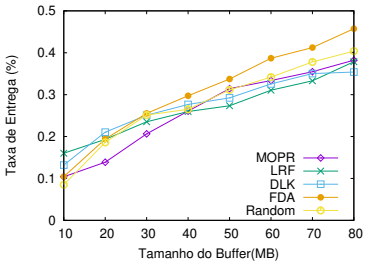


(b) Atraso médio de entrega.

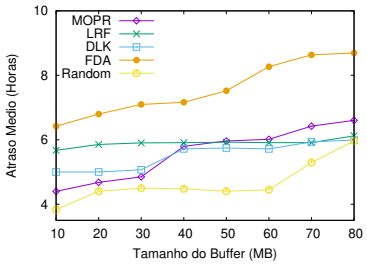


(c) Custo médio de encaminhamentos.

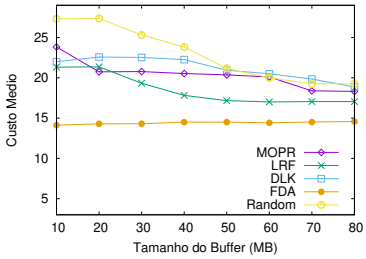
Figura 5.1: Resultados do algoritmo Epidêmico no cenário Sassy.



(a) Taxa de entrega.



(b) Atraso médio de entrega.



(c) Custo médio de encaminhamentos.

Figura 5.2: Resultados do algoritmo Epidêmico no cenário Reality.

5.4.2 Resultados com roteamento FSF

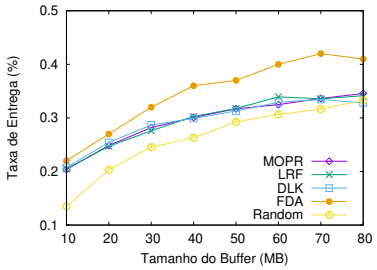
O algoritmo FSF, diferentemente do algoritmo roteamento Epidêmico, atua de forma seletiva, isto é, envia mensagens apenas para os nós que satisfazem um determinado critério. Os critérios usados por FSF para encaminhamento de mensagens são os relacionamentos sociais entre os nós e o nível de egoísmo do candidato a *relay* da mensagem. Evidentemente, esse caráter mais seletivo contribui para uma clara diminuição do nível de congestionamento na rede. Dessa forma, visando aumentar o nível de congestionamento na rede, as mensagens criadas nas simulações realizadas com o algoritmo FSF são de tamanhos maiores que aquelas utilizadas nas simulações com o algoritmo Epidêmico, como mostra a Tabela 5.2.

As Figuras 5.3. (A) e 5.4. (A) apresentam os resultados obtidos para a métrica taxa de entrega nos cenários avaliados. Pode-se notar que a combinação de FSF com o algoritmo FDA alcança os melhores resultados em ambos os cenários testados. Os resultados também demonstram que o uso dos relacionamentos sociais como critério para o gerenciamento de *buffer* pode contribuir para melhorar o desempenho geral da rede. Esses resultados poderiam ser melhorados caso as decisões tomadas por ambos os algoritmos (FSF e FDA) utilizassem a mesma lista de relacionamentos sociais. Observe que, para construir os relacionamentos sociais entre os nós, FSF usa uma heurística baseada em técnicas de aprendizado de máquinas combinadas com dados derivados de experimentos reais, enquanto FDA toma suas decisões levando em consideração os

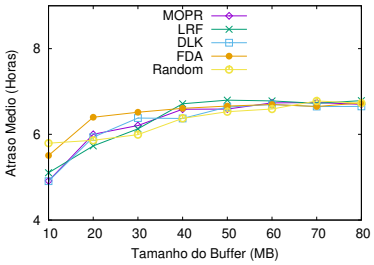
relacionamentos sociais reportados pelos próprios nós. Essa questão de implementação pode levar claramente a cenários em que a próxima mensagem que FSF gostaria de enviar não é a mesma que FDA mantém no *buffer*, fato que pode contribuir para diminuir a probabilidade de entrega.

As Figuras 5.3. (C) e 5.4. (C) apresentam os resultados da média de encaminhamentos. A partir dos resultados, nota-se que no cenário Sassy, FDA alcançou o melhor resultado, enquanto no cenário *Reality*, o algoritmo proposto alcançou um resultado muito similar a outros algoritmos testados. Os resultados alcançados por todos os algoritmos testados são fortemente influenciados pela estratégia de encaminhamento seguida por FSF. Por esse motivo, na maior parte do tempo, eles alcançaram uma média muito próxima de um encaminhamento necessário por mensagem entregue.

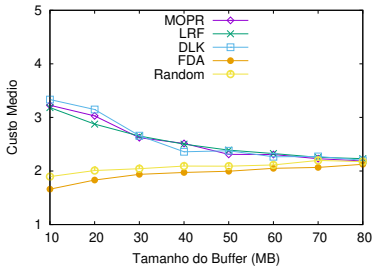
Figuras 5.3.(B) e 5.4.(B) apresentam os resultados para a métrica atraso médio de entrega. Da mesma forma que os resultados obtidos nos testes realizados com o algoritmo Epidêmico, pode-se notar que FDA foi superado por outros algoritmos testados. No entanto, novamente, a estratégia usada por FSF tem um impacto no desempenho em todos os algoritmos testados, aumentando o atraso médio.



(a) Taxa de entrega.



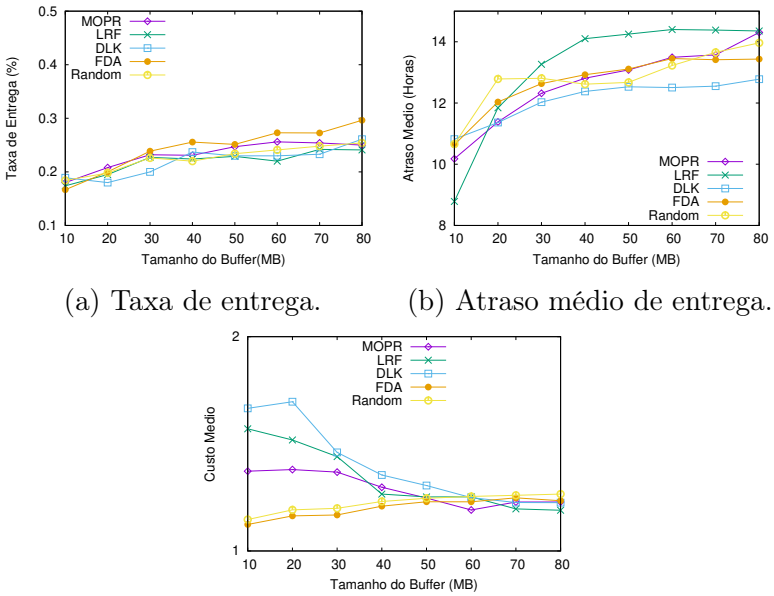
(b) Atraso médio de entrega.



(c) Custo médio de encaminhamentos.

Figura 5.3: Resultados do algoritmo FSF no cenário Sassy.

Finalmente, comparam-se os resultados obtidos pelas combinações testadas dos algoritmos de roteamento Epidêmico e FSF com o algoritmo FDA. Pode-se notar que a combinação de FDA com o algoritmo Epidêmico alcança uma melhor taxa de entrega em ambos os cenários avaliados. Isso ocorre principalmente porque o algoritmo



(a) Taxa de entrega.

(b) Atraso médio de entrega.

(c) Custo médio de encaminhamentos.

Figura 5.4: Resultados do algoritmo FSF no cenário Reality.

Epidêmico utiliza uma estratégia de replicação, a qual aumenta a probabilidade de entrega. No entanto, essa estratégia também aumenta a quantidade de recursos consumidos na rede. Dessa forma, com base nos resultados alcançados, é razoável supor que o algoritmo FDA contribua para diminuir os recursos consumidos pelo roteamento Epidêmico, como mostrado nos resultados para a métrica média de encaminhamentos.

Por outro lado, a combinação de FDA com o algoritmo FSF alcança os melhores resultados para a métrica média de encaminhamentos. Esse resultado é consequência da estratégia seguida por ambos os algoritmos FSF e FDA. Naturalmente, a estratégia seguida por FSF já diminui a quantidade média de encaminhamentos necessários para que uma mensagem alcance seu destinatário. FDA por sua vez, contribui para diminuir esse índice mantendo no *buffer* mensagens que tem uma probabilidade maior de serem utilizadas por FSF de maneira satisfatória, ou seja, que tem uma chance maior de serem entregues ao destinatário.

5.5 Considerações finais

Neste Capítulo, investigou-se o uso de relações sociais no problema de gerenciamento de *buffer* em OppNets. Dessa forma, o algoritmo Friendly-Drop foi proposto, o qual toma suas decisões de gerenciamento baseando-se nos relacionamentos sociais existentes entre os nós. Quando surge uma oportunidade de contato, o algoritmo FDA primeiro

envia as mensagens endereçadas aos nós que são amigos ou que pertencem ao grupo de nós conhecidos do nó que está tomando a decisão. Por outro lado, quando o *buffer* dos nós opera perto de sua capacidade, FDA descarta primeiramente as mensagens endereçadas a nós desconhecidos do nó que toma a decisão.

Para validar o algoritmo proposto, foram realizados um conjunto de experimentos com objetivo de determinar a eficácia da entrega de mensagens em dois cenários com base em simulações controladas por traces usando o simulador The ONE. Comparando-se com outros quatro algoritmos da mesma categoria, o algoritmo FDA forneceu melhores resultados em relação a um conjunto de métricas padrão, ou seja, a taxa de entrega e o custo médio, ao mesmo tempo, em que forneceu valores semelhantes aos dos demais algoritmos com relação ao atraso médio de entrega. Com base nesses resultados, conclui-se que de fato os relacionamentos sociais são um interessante critério a ser utilizado no problema do gerenciamento de *buffer* dos nós em OppNets, contribuindo de maneira satisfatória para a melhoria do desempenho geral da rede.

Publicação	**Idéia principal**
[81]	Divide as mensagens em três filas de acordo com informações globais sobre a rede: uma fila de mensagens prioritárias, uma fila de mensagens de média prioridade, e uma fila de mensagens de baixa prioridade. Quando o *buffer* está cheio, descarta mensagens com menos efeito no desempenho da rede (aquelas que pertencem à fila de mensagens com baixo peso).
[43]	Busca maximizar a ocupação do *buffer* através da otimização de um parâmetro T, que representa a diferença de tamanho entre a última mensagem recebida e o espaço disponível em *buffer*. Em seguida, descarta as mensagens de *buffer* de tamanho igual a T.
[18]	Utiliza a probabilidade de que qualquer mensagem distinta possa ser entregue a vários destinos antes de seu parâmetro MDR expirar. Mensagens com maior valor de MDR devem ser replicadas primeiro, e aquelas com o menor MDR podem ser descartadas primeiro.
[65]	Baseia-se em algumas características do estado da rede, tais como número de encaminhamentos e TTL das mensagens. Mensagens com maior número de encaminhamentos e menor TTL são descartadas primeiro.
[26]	Calcula o valor da utilidade de cada mensagem com objetivo de minimizar o atraso médio de entrega.
[13]	Esse algoritmo toma as suas decisões usando uma avaliação do impacto de descartar cada mensagem armazenada em *buffer* de acordo com as informações de rede coletadas para obter uma relação de entrega de mensagem ótima ou menos atraso na entrega da mensagem.
[25]	Esse algoritmo inclui uma função de utilidade que requer informações de rede global para realizar o descarte de mensagens.
[25]	Esse algoritmo usa o histórico da rede para estimar o estado atual dos parâmetros de rede necessários (globais) e usa essas estimativas, em vez de valores reais, para calcular os utilitários de mensagens para cada métrica de meta de desempenho.
[18]	Considerando algumas desvantagens de [25], nesse trabalho os autores propõem o algoritmo E-GBSD. As principais contribuições são a formulação da probabilidade de entrega para múltiplos destinos em relação a qualquer mensagem, e uma nova função de utilidade para priorizar mensagens em um *buffer* local.
[61]	Esse algoritmo toma as decisões de gerenciamento de *buffer* utilizando propriedades de mensagens, como o número total estimado de réplicas, o tempo decorrido e TTL.
[55]	Os autores propõem um algoritmo de gerenciamento de *buffer* baseado no Processo de Decisão de Markov Parcialmente Observado (POMDP). A estrutura do POMDP é empregada considerando o número de transmissões de mensagens e o nível de cooperação dos nós na rede. Essas informações ajudam o algoritmo a saber quais estratégias adotar e quais ações tomar para maximizar sua recompensa média.
[27]	Esse algoritmo de gerenciamento de *buffer* utiliza uma função de utilidade, a qual é formulada usando informações de rede global, restrições de recursos, mobilidade dos nós, e tamanhos de mensagens variados.

Tabela 5.1: Um resumo dos principais algoritmos de gerenciamento de *buffer* propostos nos últimos 5 anos para OppNets.

Tabela 5.2: Parâmetros utilizados nas simulações.

Taxa de geração de mensagens	1000 msg por simulação
Tamanho da mensagem - Epidêmico	1-2 MB
Tamanho de mensagem - FSF	2-5 MB
TTL Epidêmico	5h
TTL FSF	48h
FSF α	30%
FSF β	70%
Interface de rede	Bluetooth
Quantidade de nós	27 (Sassy) - 97 (Reality)
Duração dos traces	79 days (Sassy) - 246 days (Reality)

Capítulo 6

Compreendendo o desempenho de FSF

No Capítulo 4 da presente tese de doutorado apresentou-se o algoritmo FSF, um algoritmo de roteamento para OppNets que toma as suas decisões de roteamento considerando os relacionamentos sociais entre os nós da rede e o nível de egoísmo do nó candidato a *relay*. Os resultados obtidos nas avaliações de desempenho realizadas mostraram que em cenários de rede com a presença de nós egoístas o algoritmo FSF superou os demais algoritmos testados, fornecendo melhores índices de taxa de entrega e de custo médio de encaminhamento, além de um razoável resultado para o atraso médio de entrega.

O presente Capítulo apresenta um estudo realizado especificamente sobre o desempenho do algoritmo FSF. O principal objetivo é identificar pontos de melhorias que podem ser abordados em novas versões do algoritmo. Dessa forma, para alcançar esse objetivo implementações alternativas do algoritmo FSF foram propostas, desenvolvidas e avaliadas nos cenários de mobilidade citados na Tabela 3.2.

6.1 Implementações alternativas testadas

Para alcançar o objetivo citado acima, três implementações alternativas do algoritmo FSF foram propostas:

- (i) ***FSF with social-based Buffer Management (FSF + BM)*** - o objetivo da implementação dessa variante de FSF é avaliar o impacto do uso de um algoritmo de gerenciamento de *buffer* que favoreça as decisões de encaminhamento

de FSF. Nas avaliações de desempenho realizadas no Capítulo 4 utilizou-se como algoritmo de gerenciamento de *buffer* o algoritmo *Drop Oldest*.

De acordo com [76], o escalonamento e o descarte de mensagens podem impactar fortemente o desempenho do roteamento em Redes Oportunistas. A intuição é muito simples: quando o *buffer* dos nós é limitado, rapidamente os nós podem operar próximo de sua capacidade de armazenamento. Dessa forma, quando existe uma nova mensagem para ser recebida, mas não há espaço suficiente para armazená-la, é necessário que uma ou mais mensagens sejam retiradas do *buffer*. Nesse caso, a melhor decisão é preservar as mensagens que tenham mais chances de serem entregues em contatos futuros. Além disso, quando surge uma oportunidade de contato, os nós precisam decidir quais mensagens serão enviadas primeiro. Essa questão também é de suma importância porque a duração do contato pode não ser suficiente para que os nós troquem entre si todas as mensagens que eles desejam. O algoritmo escolhido foi o algoritmo *Friendly-Drop* proposto no Capítulo 6, o qual também é baseado em relacionamentos sociais.

- (ii) ***FSF without Energy Constraint (FSFwEC)*** - o objetivo é analisar o impacto no desempenho do algoritmo FSF causado por eventos de restrição de energia. Observe que quanto mais contatos/eventos de encaminhamento existirem, mais energia é consumida por cada nó. Dessa forma, os nós podem rapidamente operar sem energia suficiente para participar da rede. Neste cenário o algoritmo *Drop Oldest* é usado como gerenciamento do *buffer* dos nós.

- (iii) ***FSF without Selfish Nodes (FSFwSN)*** - o objetivo é analisar o impacto no desempenho do algoritmo FSF quando não existem nós egoístas na rede. Diferentemente da implementação FSFwEC, neste cenário considera-se que não há nós se comportando de maneira egoísta na rede. No entanto, eles podem se recusar a receber mensagens se seus dispositivos estiverem com restrições de recursos. Neste cenário o algoritmo *Drop Oldest* é usado como gerenciamento do *buffer* dos nós.

6.2 Resultados obtidos

Para avaliar o desempenho das variações de FSF implementadas, utilizaram-se simulações baseadas em traces. Os traces de mobilidade escolhidos são os mesmos citados na Tabela 3.2. As métricas de interesse são as mesmas citadas na Seção 3.7. Nas simulações utilizou-se a distribuição uniforme para gerar 1000 mensagens entre pares de nós.

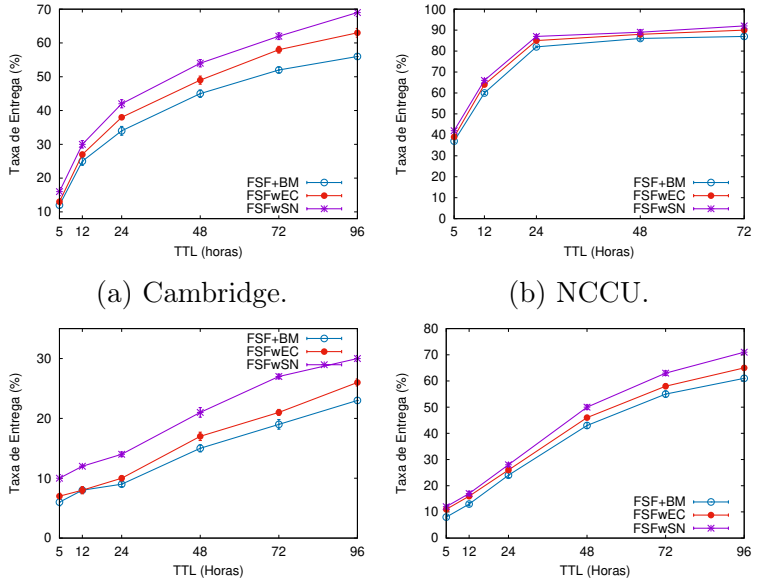
Em cada experimento realizado construiu-se para cada resultado obtido um intervalo de confiança de 95 % para os parâmetros estimados. O tamanho da mensagem varia de 512 KB a 1 MB, enquanto o tamanho do *buffer* varia de 10 MB a 60 MB, e os valores de TTL variam de 5 a 96 horas.

6.2.1 Resultados para variação de TTL

As Figuras 6.1, 6.2, e 6.3 apresentam os resultados variando-se o TTL da mensagem nos quatro cenários de mobilidade testados. Observe que, com relação à taxa de entrega, nesses cenários, todas as combinações testadas obtiveram o mesmo comportamento: contribuiram para o aumento da taxa de entrega quando comparadas a implementação original do algoritmo FSF. Também pode-se observar que em todos os cenários a ausência de nós egoístas alcançou a melhor taxa de entrega. Vale a pena mencionar que quando os nós não consideram as restrições de nível de energia, mais mensagens são entregues na rede. Ou seja, é razoável supor que várias mensagens não são entregues porque os nós operam próximos de sua capacidade energética. Por outro lado, apesar de FSF+BM ter alcançado o menor resultado em relação às demais variantes testadas, percebe-se um aumento da taxa de entrega na rede quando comparado à implementação original do algoritmo FSF.

Com relação ao atraso na entrega e ao custo médio, todas as implementações testadas obtiveram o mesmo comportamento: quanto maior o TTL da mensagem, maior o atraso médio de entrega. Com relação ao atraso de entrega, os resultados obtidos apresentam um aumento no atraso médio de entrega de todas as implementações realizadas comparando-se com a implementação original (por exemplo, cenários de Cambridge e NCCU). Uma justificativa pode ser o aumento da taxa de entrega, ou seja, quanto maior a taxa de entrega, maior será o atraso médio de entrega.

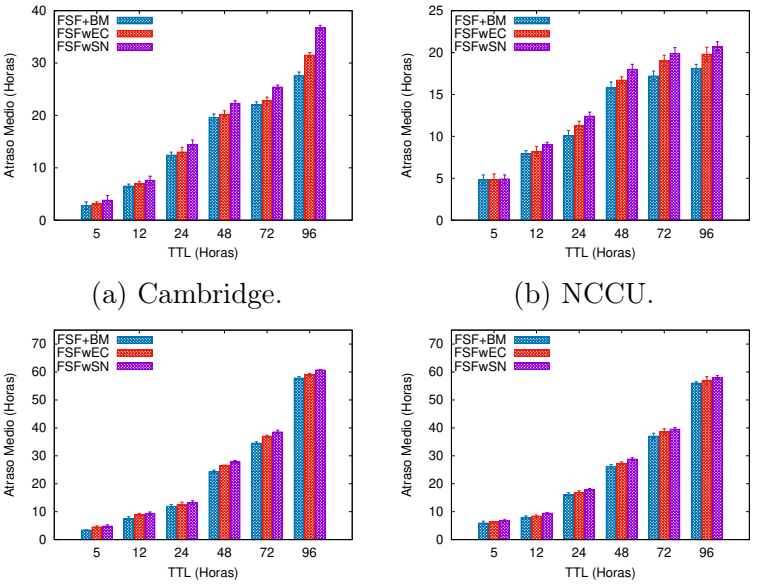
Para o custo médio, todas as implementações realizadas também aumentam o número de encaminhamentos necessários para entregar uma mensagem com sucesso. O custo médio é maior quando não se considerou a presença de nós egoístas na rede. Ou seja, é razoável supor que na implementação original de FSF, uma quantidade razoável de nós é selecionada como *relay*, mas eles se recusam a receber a mensagem porque são egoístas. Consequentemente, o número de encaminhamentos diminui, assim como a probabilidade de entrega da mensagem.



(a) Cambridge. (b) NCCU.

(c) Sassy. (d) Reality.

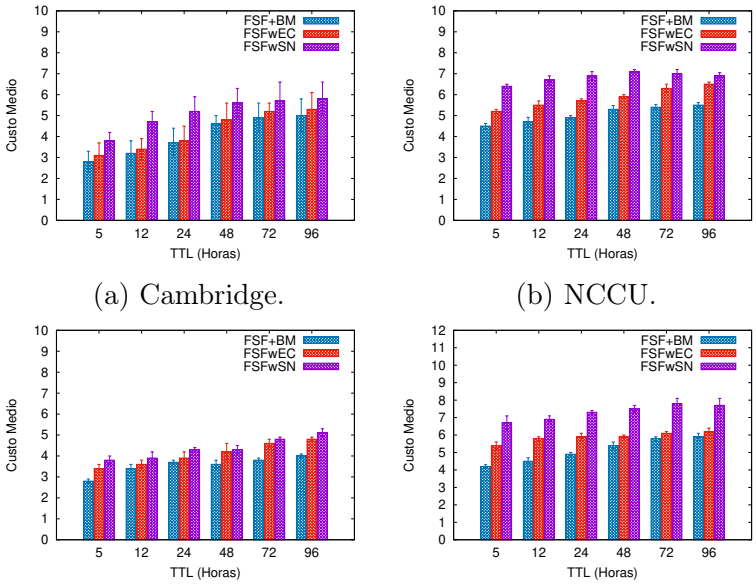
Figura 6.1: Taxa de entrega ao variar o tamanho do TTL das mensagens.



(a) Cambridge. (b) NCCU.

(c) Sassy. (d) Reality.

Figura 6.2: Atraso médio ao variar o tamanho do TTL das mensagens.



(a) Cambridge. (b) NCCU.

(c) Sassy. (d) Reality.

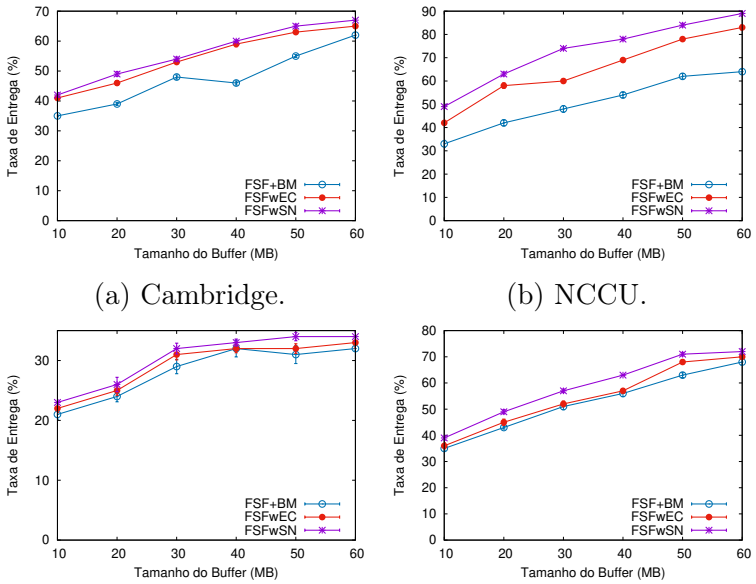
Figura 6.3: Custo médio de encaminhamentos ao variar o tamanho do TTL das mensagens.

6.2.2 Resultados para variação de buffer

As Figuras 6.4, 6.5, e 6.6 apresentam os resultados variando-se o tamanho do *buffer*. Mais uma vez, a variante FSFwSN alcançou o melhor resultado com relação à taxa de entrega. Além disso, a variante FSFwEC superou a variante FSF+BM. No entanto, todas as implementações testadas alcançaram uma taxa de entrega maior quando comparados a implementação original de FSF. É interessante destacar que em alguns cenários, como *Reality* e Sassy, pode-se observar que FSFwSN alcançou um desempenho muito próximo da implementação FSFwEC.

Com relação ao atraso médio de entrega e ao custo médio, mais uma vez o comportamento é semelhante: quanto maior o tamanho do *buffer*, maior o atraso de entrega/custo médio. Mais uma vez a variante FSFwSN superou as demais e obteve o melhor resultado tanto para o atraso médio de entrega, quanto para o custo médio.

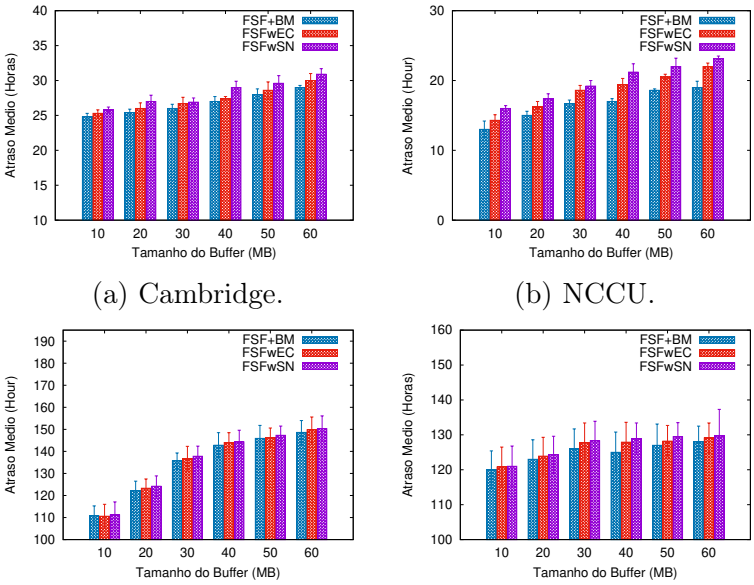
Esses resultados demonstram que o desempenho de FSF, assim como outros algoritmos propostos na literatura, são fortemente impactados pela presença de nós egoístas na rede. No entanto, considerando os resultados obtidos pela variante FSF+BM é razoável supor que alguns fatores como o gerenciamento de recursos como o *buffer* e a energia dos nós podem ajudar a minimizar o impacto da presença de nós egoístas na rede.



(a) Cambridge. (b) NCCU.

(c) Sassy. (d) Reality.

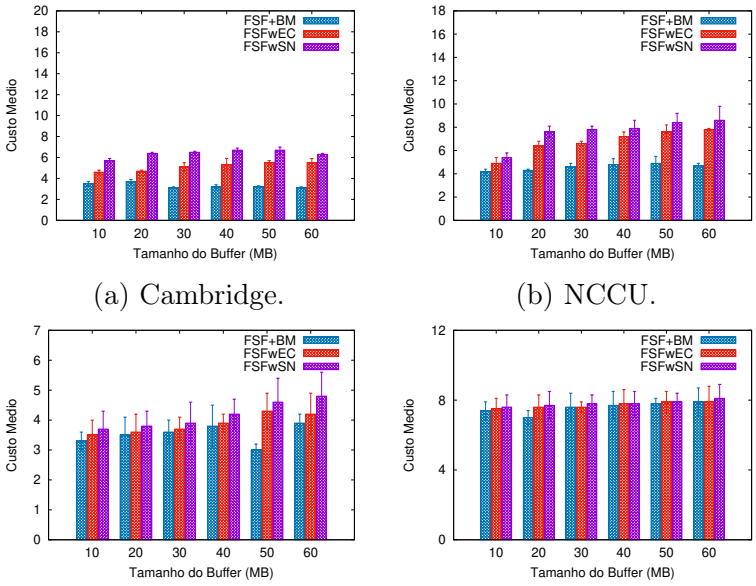
Figura 6.4: Taxar de entrega ao variar o tamanho do buffer dos nós.



(a) Cambridge. (b) NCCU.

(c) Sassy. (d) Reality.

Figura 6.5: Atraso médio ao variar o tamanho do buffer dos nós.



(a) Cambridge. (b) NCCU.

(c) Sassy. (d) Reality.

Figura 6.6: Custo médio de encaminhamentos ao variar o tamanho do buffer dos nós.

6.3 Considerações Finais

No presente Capítulo, realizou-se um estudo mais aprofundado sobre o desempenho do algoritmo FSF. O principal objetivo foi identificar as variáveis que causam um impacto maior no desempenho do algoritmo. Dessa forma, os pontos de melhorias que podem ser identificados e trabalhados em novas versões do algoritmo. Para alcançar esse objetivo implementações alternativas do algoritmo FSF foram propostas, desenvolvidas e avaliadas nos cenários de mobilidade citados na Tabela 3.2.

A partir dos resultados obtidos, pode-se concluir que da mesma maneira que em outros trabalhos na literatura, a presença de nós egoístas nas simulações realizadas degrada o desempenho da rede de maneira geral. É razoável concluir também que o impacto causado pela falta de recursos nos dispositivos dos nós causa um impacto considerável, mas esse problema pode ser mitigado por uma diminuição da quantidade de eventos de encaminhamentos/escaneamentos realizados pelos nós. Finalmente, os resultados obtidos pela combinação entre algoritmo de roteamento e de gerenciamento de recursos baseados em relacionamentos sociais pode fornecer resultados próximos dos obtidos quando da ausência de nós egoístas na rede. Dessa forma, em implementações futuras o algoritmo de gerenciamento de *buffer* pode se tornar um fator crucial para a melhoria do desempenho do algoritmo FSF.

Capítulo 7

Conclusões e trabalhos futuros

Essa tese de doutorado apresentou um estudo visando avaliar o uso dos relacionamentos sociais como critério para a disseminação de mensagens em OppNets, que culminou com o projeto, a implementação e a avaliação dos seguintes algoritmos:

- Friendship and Selfishness Forwarding (FSF) – algoritmo de roteamento que considera duas características sociais: os relacionamentos sociais entre os nós e o egoísmo dos nós candidatos a *relay*. FSF realiza duas tarefas principais: em primeiro lugar, FSF classifica os laços sociais entre os nós; segundo, FSF usa um sistema de reputação para verificar o egoísmo do nó *relay* considerando os casos em que, apesar de uma forte relação social com o destino, o nó *relay* pode recusar o recebimento da mensagem porque ele é egoísta ou seu dispositivo possui restrições de recursos naquele instante.

- Friendly-Drop Algorithm (FDA) – um algoritmo de gerenciamento de *buffer* que considera os relacionamentos sociais entre os nós para tomar as decisões de descarte/envio de mensagens. Quando ocorre o evento do *overflow* no *buffer* de um nó, FDA descarta primeiramente mensagens destinadas a usuários desconhecidos do nó que toma a decisão de gerenciamento. Por outro lado, FDA ordena a fila de mensagens a serem encaminhadas em uma oportunidade de contato priorizando as mensagens destinadas aos nós pertencentes ao grupo de amigos do nó que toma a decisão.

Para validar o algoritmo FSF, realizou-se um conjunto de experimentos para determinar a eficácia da entrega de mensagens em quatro cenários baseados em simulações controladas por traces usando o simulador *The ONE*. Em comparação com outros algoritmos de roteamento da mesma classe, FSF ofereceu melhores resultados em relação

a um conjunto de métricas padrão, ou seja, a taxa de entrega, o custo médio, além de alcançar resultados razoáveis de atraso de entrega. Também realizou-se um conjunto de experimentos para determinar o impacto de três cenários alternativos. A partir dos resultados obtidos, pode-se observar que o desempenho de FSF é fortemente impactado pela presença de nós egoístas na rede. No entanto, também pode-se concluir a partir desses resultados que o algoritmo de gerenciamento de *buffer* pode contribuir para aumentar a taxa de entrega, mesmo em cenários com a presença de nós egoístas na rede.

Para validar o algoritmo FDA, foi realizado um conjunto de experimentos com objetivo de determinar a eficácia da entrega de mensagens em dois cenários com base em simulações controladas por traces usando o simulador The ONE. Comparando-se com outros quatro algoritmos da mesma categoria, o algoritmo FDA forneceu melhores resultados em relação a um conjunto de métricas padrão, ou seja, a taxa de entrega e o custo médio, ao mesmo tempo, em que forneceu valores semelhantes aos dos demais algoritmos com relação ao atraso médio de entrega. Com base nesses resultados, conclui-se que de fato os relacionamentos sociais são um interessante critério a ser utilizado no problema do gerenciamento de *buffer* dos nós em OppNets, contribuindo de maneira satisfatória para a melhoria do desempenho geral da rede.

Finalmente, com objetivo de identificar pontos de melhoria no algoritmo FSF, realizou-se um estudo aprofundado de implementações alternativas desse algoritmo. Três variantes foram testadas:

- FSF+BM – uma variante de FSF que utiliza como algoritmo de gerenciamento de *buffer* o algoritmo FDA, que também toma suas decisões considerando os relacionamentos sociais entre os nós.

- FSFwEC –uma variante de FSF que desconsidera as questões relacionadas as limitações de energia dos dispositivos que compõem a rede.

- FSFwSN –uma variante de FSF que desconsidera a presença de nós egoístas na rede.

Uma avaliação de desempenho utilizando quatro cenários de mobilidade real foi realizada. A partir dos resultados obtidos, concluiu-se que da mesma maneira que em outros trabalhos na literatura, a presença de nós egoístas nas simulações realizadas degradou o desempenho do algoritmo FSF e da rede de maneira geral. Concluiu-se também que a falta de recursos nos dispositivos dos nós causa um impacto considerável no desempenho de FSF, porém esse problema pode ser mitigado por uma diminuição

da quantidade de eventos de encaminhamentos/escaneamentos realizados pelos nós. Finalmente, os resultados obtidos pela combinação entre algoritmo de roteamento e de gerenciamento de recursos baseados em relacionamentos sociais pode fornecer resultados próximos dos obtidos quando da ausência de nós egoístas na rede. Dessa forma, em implementações futuras o algoritmo de gerenciamento de *buffer* pode ser tornar um fator crucial para a melhoria do desempenho do algoritmo FSF.

Como trabalho futuro e continuação da investigação iniciada na presente tese de doutorado, pretende-se avaliar as técnicas propostas e apresentadas no presente documento em algoritmos para o gerenciamento de energia dos nós da rede. Também pretende-se realizar um estudo comparativo da adoção de outros algoritmos de aprendizado de máquina para classificar os relacionamentos sociais entre os nós. Finalmente, pretende-se analisar o impacto do egoísmo no processo de roteamento usando diferentes limites para restrições de recursos de dispositivos.

Capítulo 8

Publicações

Os resultados obtidos na execução da investigação realizada na presente tese de doutorado foram publicados nos seguintes eventos:

- Souza, C., Mota, E., Galvao, L., Manzoni, P., & Cano, J. C. (2014, September). **Drop less known strategy for buffer management in dtn nodes.** In Proceedings of the Latin America Networking Conference on LANC 2014 (p. 6). ACM. [68]

- Souza, C., Mota, E., Galvao, L., Manzoni, P., Cano, J. C., & Calafate, C. T. (2016, June). **Fsf: Friendship and selfishness forwarding for delay tolerant networks.** In Computers and Communication (ISCC), 2016 IEEE Symposium on (pp. 1200-1207). IEEE. [69]

- Souza, C., Mota, E., Manzoni, P., Cano, J. C., & Calafate, C. T. (2016, August). **Improving delivery delay in social-based message forwarding in Delay Tolerant Networks.** In Proceedings of the 2016 workshop on Fostering Latin-American Research in Data Communication Networks (pp. 52-54). ACM. [70]

- Souza, C. B., Mota, E., Galvão, L., & Soares, D. (2017, May). **Gerenciamento de Buffer Baseado em Egoísmo para Redes Tolerantes a Atrasos e Desconexões.** In Simpósio Brasileiro de Redes de Computadores (SBRC) (Vol. 35, No. 1/2017). [73]

- Souza, C., Mota, E., Manzoni, P., Cano, J. C., Calafate, C. T., Hernández-Orallo, E., & Tapia, J. H. (2018, April). **Friendly-drop: A social-based buffer management algorithm for opportunistic networks.** In Wireless Days (WD), 2018 (pp. 172-177). IEEE. [71]

Estes artigos foram estendidos e deram origem aos seguintes trabalhos publicados nas seguintes revistas:

- Souza, C., Mota, E., Soares, D., Manzoni, P., Cano, J. C., & Calafate, C. T. **FSF: Applying Machine Learning Techniques to Data Forwarding in Socially Selfish Opportunistic Networks**. Sensors, v. 19, n. 10, p. 2374, 2019. [72]

- Vegni, A., Loscri, V., Manzoni, P., Hernández, E., Souza, C. (2018). **Data Transmissions using Hub Nodes in Vehicular Social Networks.** (IEEE TRANSACTIONS ON MOBILE COMPUTING). TMC v. 9, p. 1-1, 2019. [80]

- Souza, C., Mota, E., Galvao, L., Manzoni, P., Cano, J. C., & Calafate, C. T. (2019). **Exploiting social relationships for efficient resources management decisions in Opportunistic Networks.** (To be defined).

Referências Bibliográficas

[1] (2013). Projeto internet interplanetaria.

[2] Ahmed, K. K.; Omar, M. H. & Hassan, S. (2016). Routing strategies and buffer management in delay tolerant networks. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 8(10):139--143.

[3] Akestoridis, D.-G. (2016). CRAWDAD dataset uoi/haggle (v. 2016-08-28): derived from cambridge/haggle (v. 2009-05-29). Downloaded from `https://crawdad.org/uoi/haggle/20160828`.

[4] Becker, V. (2015). O impacto das mídias digitais na televisão brasileira: queda da audiência e aumento do faturamento. *Palabra-Clave*, 18(2):341--373.

[5] Bigwood, G.; Abdesslem, F. B. & Henderson, T. (2012). Predicting location-sharing privacy preferences in social network applications. *Proc. of AwareCast*, 12:1--12.

[6] Bulut, E. & Szymanski, B. K. (2010a). Friendship based routing in delay tolerant mobile social networks. Em *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pp. 1--5. IEEE.

[7] Bulut, E. & Szymanski, B. K. (2010b). Friendship based routing in delay tolerant mobile social networks. Em *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pp. 1--5. IEEE.

[8] Burgess, J.; Gallagher, B.; Jensen, D. & Levine, B. N. (2006). Maxprop: Routing for vehicle-based disruption-tolerant networks. Em *Proc. ieee infocom*, volume 6, pp. 1--11. Barcelona, Spain.

[9] Burns, B.; Brock, O. & Levine, B. N. (2008). Mora routing and capacity building in disruption-tolerant networks. *Ad hoc networks*, 6(4):600--620.

[10] Cao, Y. & Sun, Z. (2013). Routing in delay/disruption tolerant networks: A taxonomy, survey and challenges. *IEEE Communications surveys & tutorials*, 15(2):654--677.

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

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

[13] Elwhishi, A.; Ho, P.-H.; Naik, K. & Shihada, B. (2013). A novel message scheduling framework for delay tolerant networks routing. *ieee transactions on parallel and distributed systems*, 24(5):871--880.

[14] 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.

[15] 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.

[16] 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.

[17] Hernandez-Orallo, E.; Olmos, M. D. S.; Cano, J.-C.; Calafate, C. T. & Manzoni, P. (2015). Cocowa: A collaborative contact-based watchdog for detecting selfish nodes. *IEEE transactions on mobile computing*, 14(6):1162--1175.

[18] Hsu, Y.-F. & Hu, C.-L. (2016). Enhanced buffer management for data delivery to multiple destinations in dtns. *IEEE Transactions on Vehicular Technology*, 65(10):8735--8739.

[19] Hui, P.; Chaintreau, A.; Scott, J.; Gass, R.; Crowcroft, J. & Diot, C. (2005). Pocket switched networks and human mobility in conference environments. Em *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pp. 244--251. ACM.

[20] Hui, P.; Crowcroft, J. & Yoneki, E. (2011). Bubble rap: Social-based forwarding in delay-tolerant networks. *Mobile Computing, IEEE Transactions on*, 10(11):1576--1589.

[21] 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. Em *ACM Sigplan Notices*, volume 37, pp. 96--107. ACM.

[22] 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.

[23] Khabbaz, M. J.; Assi, C. M. & Fawaz, W. F. (2012). Disruption-tolerant networking: A comprehensive survey on recent developments and persisting challenges. *IEEE Communications Surveys & Tutorials*, 14(2):607--640.

[24] 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.

[25] Krifa, A.; Barakat, C. & Spyropoulos, T. (2012). Message drop and scheduling in dtns: Theory and practice. *IEEE Transactions on Mobile Computing*, 11(9):1470--1483.

[26] Le, T.; Kalantarian, H. & Gerla, M. (2016a). A buffer management strategy based on power-law distributed contacts in delay tolerant networks. Em *Computer Communication and Networks (ICCCN), 2016 25th International Conference on*, pp. 1--8. IEEE.

[27] Le, T.; Kalantarian, H. & Gerla, M. (2016b). A joint relay selection and buffer management scheme for delivery rate optimization in dtns. Em *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2016 IEEE 17th International Symposium on A*, pp. 1--9. IEEE.

[28] Li, F. & Wu, J. (2009). Localcom: a community-based epidemic forwarding scheme in disruption-tolerant networks. Em *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON'09. 6th Annual IEEE Communications Society Conference on*, pp. 1--9. IEEE.

[29] Li, L.; Qin, Y. & Zhong, X. (2016). A novel routing scheme for resource-constraint opportunistic networks: A cooperative multiplayer bargaining game approach. *IEEE Transactions on Vehicular Technology*, 65(8):6547--6561.

[30] Li, Q.; Gao, W.; Zhu, S. & Cao, G. (2012). A routing protocol for socially selfish delay tolerant networks. *Ad Hoc Networks*, 10(8):1619--1632.

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

[32] Li, W. & Joshi, A. (2009). Outlier detection in ad hoc networks using dempster-shafer theory. Em *Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on*, pp. 112--121. IEEE.

[33] Li, Y.; Hui, P.; Jin, D.; Su, L. & Zeng, L. (2010b). Evaluating the impact of social selfishness on the epidemic routing in delay tolerant networks. *Communications Letters, IEEE*, 14(11):1026--1028.

[34] Li, Y.; Qian, M.; Jin, D.; Su, L. & Zeng, L. (2009). Adaptive optimal buffer management policies for realistic dtn. Em *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pp. 1--5. IEEE.

[35] Li, Y.; Su, G.; Wu, D. O.; Jin, D.; Su, L. & Zeng, L. (2011). The impact of node selfishness on multicasting in delay tolerant networks. *Vehicular Technology, IEEE Transactions on*, 60(5):2224--2238.

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

[37] 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.

[38] Liu, C. & Wu, J. (2009). An optimal probabilistic forwarding protocolin delay tolerant networks. Em *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, pp. 105--114. ACM.

[39] Lu, X.; Lio, P. & Hui, P. (2016). Distance-based opportunistic mobile data offloading. *Sensors*, 16(6):878.

[40] Marti, S.; Giuli, T. J.; Lai, K. & Baker, M. (2000). Mitigating routing misbehavior in mobile ad hoc networks. Em *Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 255--265. ACM.

[41] Mei, A. & Stefa, J. (2012). Give2get: Forwarding in social mobile wireless networks of selfish individuals. *IEEE Transactions on Dependable and Secure Computing*, 9(4):569--582.

[42] Miao, J.; Hasan, O.; Mokhtar, S. B.; Brunie, L. & Yim, K. (2013). An investigation on the unwillingness of nodes to participate in mobile delay tolerant network routing. *International Journal of Information Management*, 33(2):252--262.

[43] Moetesum, M.; Hadi, F.; Imran, M.; Minhas, A. A. & Vasilakos, A. V. (2016). An adaptive and efficient buffer management scheme for resource-constrained delay tolerant networks. *Wireless networks*, 22(7):2189--2201.

[44] Nagwani, N. K. (2015). A comment on a similarity measure for text classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2589--2590.

[45] Naves, J. F.; Moraes, I. M. & Albuquerque, C. V. (2012). Lps e lrf: Politicas de gerenciamento de buffer eficientes para redes tolerantes a atrasos e desconexões. *Simpósio Brasileiro de Redes de Computadores (SBRC?12)*, pp. 293--305.

[46] 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.

[47] Nguyen, H. A.; Giordano, S. & Puiatti, A. (2007). Probabilistic routing protocol for intermittently connected mobile ad hoc network (propicman). Em *2007 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pp. 1--6. IEEE.

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

[49] 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.

[50] 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.

[51] Park, M.; Min, S.; So, S.; Oh, D.; Kim, B. & Lee, J. (2010). Mobility pattern based routing algorithm for delay/disruption tolerant networks. Em *Smart Spaces and Next Generation Wired/Wireless Networking*, pp. 275--286. Springer.

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

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

[54] Qin, Y.; Li, L.; Zhang, X. & Zhong, X. (2015). Nfcu: A new friendship-based routing with buffer management in opportunistic networks. *arXiv preprint arXiv:1501.07754*.

[55] Rahmouni, I.; El Kamili, M.; El Fenni, M. R.; Omari, L. & Kobbane, A. (2014). Optimal buffer management policies in dtns: A pomdp approach. Em *Communications (ICC), 2014 IEEE International Conference on*, pp. 94--99. IEEE.

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

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

[58] Shaghaghian, S. & Coates, M. (2015). Optimal forwarding in opportunistic delay tolerant networks with meeting rate estimations. *IEEE Transactions on Signal and Information Processing over Networks*, 1(2):104--116.

[59] Shah, R. C.; Roy, S.; Jain, S. & Brunette, W. (2003). Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks. *Ad Hoc Networks*, 1(2):215--233.

[60] Shevade, U.; Song, H. H.; Qiu, L. & Zhang, Y. (2008). Incentive-aware routing in dtns. Em *Network Protocols, 2008. ICNP 2008. IEEE International Conference on*, pp. 238--247. IEEE.

[61] Shin, K.; Kim, K. & Kim, S. (2012). Traffic management strategy for delay-tolerant networks. *Journal of Network and Computer Applications*, 35(6):1762--1770.

[62] Silva, D. R.; Costa, A. & Macedo, J. (2012). Energy impact analysis on dtn routing protocols. *ExtremeCom 2012*, pp. 1--6.

[63] Simon, H. A. (1983). Why should machines learn? Em *Machine learning*, pp. 25--37. Springer.

[64] Soares, D.; Mota, E.; Souza, C.; Manzoni, P.; Cano, J. C. & Calafate, C. (2014). A statistical learning reputation system for opportunistic networks. Em *2014 IFIP Wireless Days (WD)*, pp. 1--6. IEEE.

[65] Sobin, C. (2016). An efficient buffer management policy for dtn. *Procedia Computer Science*, 93:309--314.

[66] Socievole, A.; De Rango, F. & Marano, S. (2013). Face-to-face with facebook friends: using online friendlists for routing in opportunistic networks. Em *Personal Indoor and Mobile Radio Communications (PIMRC), 2013 IEEE 24th International Symposium on*, pp. 2989--2994. IEEE.

[67] Socievole, A. & Marano, S. (2012). Exploring user sociocentric and egocentric behaviors in online and detected social networks. Em *Future Internet Communications (BCFIC), 2012 2nd Baltic Congress on*, pp. 140--147. IEEE.

[68] Souza, C.; Mota, E.; Galvao, L.; Manzoni, P. & Cano, J. C. (2014). Drop less known strategy for buffer management in dtn nodes. Em *Proceedings of the Latin America Networking Conference on LANC 2014*, LANC '14, pp. 6:1--6:7, New York, NY, USA. ACM.

[69] Souza, C.; Mota, E.; Galvao, L.; Manzoni, P.; Cano, J. C. & Calafate, C. T. (2016a). Fsf: Friendship and selfishness forwarding for delay tolerant networks. Em *Computers and Communication (ISCC), 2016 IEEE Symposium on*, pp. 1200--1207. IEEE.

[70] Souza, C.; Mota, E.; Manzoni, P.; Cano, J. C.; Calafate, C. T. et al. (2016b). Improving delivery delay in social-based message forwarding in delay tolerant networks. Em *Proceedings of the 2016 workshop on Fostering Latin-American Research in Data Communication Networks*, pp. 52--54. ACM.

[71] Souza, C.; Mota, E.; Manzoni, P.; Cano, J. C.; Calafate, C. T.; Hernandez-Orallo, E. & Tapia, J. H. (2018). Friendly-drop: A social-based buffer management algorithm for opportunistic networks. Em *Wireless Days (WD), 2018*, pp. 172--177. IEEE.

[72] Souza, C.; Mota, E.; Soares, D.; Manzoni, P.; Cano, J.-C.; Calafate, C. T. & Hernández-Orallo, E. (2019). Fsf: Applying machine learning techniques to data forwarding in socially selfish opportunistic networks. *Sensors*, 19(10):2374.

[73] Souza, C. B.; Mota, E. & Soares, Leandro Galvao, D. S. (2017). Gerenciamento de buffer baseado em egoismo para redes tolerantes a atrasos e desconexoes. Em *XXXV Simposio Brasileiro de Redes de Computadores e Sistemas Distribuidos (SBRC)*. SBC.

[74] 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.

[75] Spyropoulos, T.; Psounis, K. & Raghavendra, C. S. (2008). Efficient routing in intermittently connected mobile networks: The single-copy case. *IEEE/ACM Transactions on Networking (ToN)*, 16(1):63--76.

[76] Tang, L.; Chai, Y.; Li, Y. & Weng, B. (2012). Buffer management policies in opportunistic networks. *Journal of Computational Information Systems*, 8(12):5149--5159.

[77] Trifunovic, S.; Kouyoumdjieva, S. T.; Distl, B.; Pajevic, L.; Karlsson, G. & Plattner, B. (2017). A decade of research in opportunistic networks: challenges, relevance, and future directions. *IEEE Communications Magazine*, 55(1):168--173.

[78] Tsai, T.-C. & Chan, H.-H. (2015). Nccu trace: social-network-aware mobility trace. *IEEE Communications Magazine*, 53(10):144--149.

[79] Vahdat, A.; Becker, D. et al. (2000). Epidemic routing for partially connected ad hoc networks. Relatório técnico, Technical Report CS-200006, Duke University.

[80] Vegni, A. M.; Souza, C.; Loscri, V.; Hernandez-Orallo, E. & Manzoni, P. (2019). Data transmissions using hub nodes in vehicular social networks. *IEEE Transactions on Mobile Computing*.

[81] Wang, H.; Wang, H.; Feng, G. & Lv, H. (2017). Nwbbmp: a novel weight-based buffer management policy for dtn routing protocols. *Peer-to-Peer Networking and Applications*, pp. 1--7.

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

[83] Witten, I. H.; Frank, E.; Hall, M. A. & Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

[84] Xia, F.; Liu, L.; Li, J.; Ma, J. & Vasilakos, A. V. (2015). Socially aware networking: A survey. *IEEE Systems Journal*, 9(3):904--921.

[85] Zeng, F.; Zhao, N. & Li, W. (2017). Effective social relationship measurement and cluster based routing in mobile opportunistic networks. *Sensors*, 17(5):1109.

[86] Zhang, L.; Wang, X.; Lu, J.; Ren, M.; Duan, Z. & Cai, Z. (2014). A novel contact prediction-based routing scheme for dtns. *Transactions on Emerging Telecommunications Technologies*.

[87] Zhao, W.; Ammar, M. & Zegura, E. (2004). A message ferrying approach for data delivery in sparse mobile ad hoc networks. Em *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pp. 187--198. ACM.

[88] Zhao, W.; Ammar, M. & Zegura, E. (2005). Controlling the mobility of multiple data transport ferries in a delay-tolerant network. Em *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 2, pp. 1407--1418. IEEE.

[89] Zhao, W. & Ammar, M. H. (2003). Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. Em *Distributed Computing Systems, 2003. FTDCS 2003. Proceedings. The Ninth IEEE Workshop on Future Trends of*, pp. 308--314. IEEE.

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

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