UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Tecnologia

LUCAS BONETTI

ANÁLISE DO DESEMPENHO DO PROTOCOLO *NETWORK-FRIENDLY EPIDEMIC MULTICAST* EM REDES SEM FIO *AD HOC*

LIMEIRA
2019

LUCAS BONETTI

ANÁLISE DO DESEMPENHO DO PROTOCOLO *NETWORK-FRIENDLY EPIDEMIC MULTICAST* EM REDES SEM FIO *AD HOC*

Dissertação apresentada à Faculdade de Tecnologia da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Tecnologia, na área de Sistemas de Informação e Comunicação.

Orientador: Prof. Dr. André Franceschi de Angelis

ESTE TRABALHO CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO LUCAS BONETTI, E ORIENTADO PELO PROF. DR. ANDRÉ FRANCESCHI DE ANGELIS.

LIMEIRA
2019

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Faculdade de Tecnologia
Felipe de Souza Bueno - CRB 8/8577

Bonetti, Lucas, 1995-
B641a Análise do desempenho do protocolo *Network-friendly Epidemic Multicast* em redes sem fio *ad hoc* / Lucas Bonetti. – Limeira, SP : [s.n.], 2019.

Orientador: André Franceschi de Angelis.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Tecnologia.

1. Redes ad hoc (Redes de computadores). 2. Sistemas distribuídos. 3. Multicasting (Redes de computadores). I. Angelis, Andre Franceschi de, 1969-. II. Universidade Estadual de Campinas. Faculdade de Tecnologia. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Performance analysis of the Network-friendly Epidemic Multicast protocol on ad hoc wireless networks
Palavras-chave em inglês:
Ad hoc networks (Computer networks)
Distributed systems
Multicasting (Computer networks)
Área de concentração: Sistemas de Informação e Comunicação
Titulação: Mestre em Tecnologia
Banca examinadora:
André Franceschi de Angelis [Orientador]
Paulo Sérgio Martins Pedro
Kalinka Regina Lucas Jaquie Castelo Branco
Data de defesa: 13-06-2019
Programa de Pós-Graduação: Tecnologia

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0003-3543-6800
- Currículo Lattes do autor: http://lattes.cnpq.br/5825418749435667

FOLHA DE APROVAÇÃO

Abaixo se apresentam os membros da comissão julgadora da sessão pública de defesa de dissertação para o Título de Mestre em Tecnologia na área de concentração de Sistemas de Informação e Comunicação, a que submeteu o aluno Lucas Bonetti, em 13 de junho de 2019 na Faculdade de Tecnologia – FT/UNICAMP, em Limeira/SP.

Prof. Dr. André Franceschi de Angelis
Presidente da Comissão Julgadora

Prof. Dr. Paulo Sérgio Martins Pedro
UNICAMP

Profa. Dra. Kalinka Regina Lucas Jaquie Castelo Branco
USP

Ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós-Graduação da Faculdade de Tecnologia.

DEDICATÓRIA

Dedico este trabalho a meu avô Luiz Bonetti (*in memoriam*).

AGRADECIMENTOS

Agradeço primeiramente a Deus pelo dom da vida, pelas oportunidades e pessoas que coloca em meu caminho, sem as quais nada seria possível. Aos meus pais, Luiz e Deise, minha eterna gratidão por valores como amor, educação e honestidade recebidos. Ao meu orientador, Professor André Franceschi de Angelis, pelos ensinamentos que levarei por toda vida. A minha noiva, Larissa Gatti, pela imensa paciência e confortante apoio em todas as situações ao longo do caminho. Aos mais que colegas de trabalho, verdadeiros amigos José Renato Paviotti, Larissa Alves, Júnio Oliveira e Washington Benício, o meu obrigado pelas incontáveis formas de apoio recebidas inúmeras vezes. Gratidão também ao Instituto Federal de São Paulo, pela bolsa de incentivo educacional e, em especial, ao Câmpus Capivari, do qual faço parte, por viabilizar os experimentos em laboratório deste trabalho.

RESUMO

Os protocolos epidêmicos surgiram como uma interessante estratégia de comunicação em sistemas distribuídos, com o poder de disseminar mensagens na rede em pouco tempo, independentemente de topologia e com relativa imunidade a falhas das estações. Porém, o custo desta propagação pode ser alto devido à inundação da rede com mensagens duplicadas. Dentre os diversos protocolos *gossip-based*, destaca-se o *Network-friendly Epidemic Multicast* (NEEM), que foi concebido com o objetivo de ser amigável à rede, utilizando recursos do *Transmisson Control Protocol* (TCP) além de uma técnica de gerenciamento de *buffer* para descarte de mensagens obsoletas. Este trabalho investigou o seu desempenho em *mobile ad hoc networks* (MANETs), redes sem fio descentralizadas e autoconfiguráveis, em um cenário realista de baixa mobilidade, contrapondo o seu uso ao emprego de técnicas consagradas de broadcast sobre o *User Datagram Protocol* (UDP). O método experimental usado foi a construção de um programa realista de suporte a atendimento a emergências, capaz de simular mobilidade e de comunicar-se via diferentes protocolos, à escolha do usuário. Este programa foi executado num ambiente controlado de 30 computadores conectados por uma MANET, registrando a troca de mensagens em arquivos (*logs*) para análise. Nas métricas consideradas, a comunicação UDP obteve melhor desempenho, indicando que o NEEM e protocolos epidêmicos de comportamento similar são inadequados para o cenário considerado.

Palavras-chave: Redes *ad hoc*, Sistemas distribuídos, *Multicasting*.

ABSTRACT

Epidemic protocols have emerged as an interesting communication strategy in distributed systems, with the power to disseminate messages on the network in a short time, independently of topology and with relative immunity to station failures. However, the cost of this communication may be high because of the flood of the network with duplicate messages. Among the various gossip-based protocols, we highlight the Network-friendly Epidemic Multicast (NEEM), which was designed to be network-friendly, using Transmisson Control Protocol (TCP) features as well as a technique of buffer management for discarding obsolete messages. This work investigated the performance of mobile ad hoc networks (MANETs), decentralized and self-configuring wireless networks, in a realistic scenario of low mobility, opposing its use to the use of broadcast techniques on the User Datagram Protocol (UDP). The experimental method used was the construction of a realistic emergency support program capable of simulating mobility and communicating via different protocols, at the user's choice. This program was run in a controlled environment of 30 computers connected by a MANET, recording the exchange of messages in files (logs) for analysis. In the considered metrics, the UDP communication obtained better performance, indicating that the NEEM and epidemic protocols of similar behavior are inadequate for the scenario considered.

Keywords: Ad hoc networks, Distributed systems, Multicasting.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AP – *Access point*

BSS – *Basic Service Set*

GHz - *Gigahertz*

ICMP - *Internet Control Message Protocol*

IP - *Internet Protocol*

ISO/OSI - *International Organization for Standardization/Open System Interconnection*

Kbps – Quilobytes por segundo

LAN – *Local Area Network*

MANET - *Mobile ad hoc network*

Mbps – Megabits por segundo

NEEM - *Network-Friendly Epidemic Multicast*

NS - *Network simulator*

NTP – *Network Time Protocol*

Pbcast - *Probabilistic Broadcast Protocol*

SGBD – Sistema gerenciador de banco de dados

SQL – *Structured Query Language*

TCP - *Transmission Control Protocol*

TTL – *Time to live*

UDP - *User Datagram Protocol*

WLAN – *Wireless Local Area Network*

SUMÁRIO

1. Introdução

As redes de computadores possibilitam acesso e troca de informações [1;2]. Uma rede pode ser definida como um conjunto de dispositivos (ou nós) conectados por *links* de comunicação. Os nós, por sua vez, são dispositivos capazes de enviar e/ou receber dados [2]. Redes são construídas a partir de um hardware programável de uso geral e tem como característica principal a generalidade. Dessa forma, são capazes de transportar muitos dados diferentes admitindo grande e crescente gama de aplicações [3]. Como critérios principais que devem caracterizar as redes podem ser citados desempenho, confiabilidade e segurança [2].

As redes podem ser classificadas de acordo com diferentes critérios, como a topologia, ou seja, a maneira como os dispositivos são conectados. Sendo assim, uma rede pode ter a topologia sem fio, não necessitando do uso de cabos para haver a conexão. No caso de determinadas redes, é usado um ponto de acesso (*Wireless Access Point*) [1]. Uma rede sem fio integra redes de dados, que permitem o compartilhamento de informações, e a comunicação de rádio ou sem fio que usa radiação eletromagnética para mover as informações de um local para outro [4].

As redes sem fio em que não existe um ponto de acesso e nas quais os dispositivos se comunicam diretamente entre si são conhecidas como redes móveis *ad hoc* (MANETs) e tornaram-se uma importante tecnologia emergente em computação móvel, sendo uma das áreas de pesquisa mais prevalentes nos últimos anos [5]. Por permitirem comunicação sem infraestrutura física, tornam-se atrativas pela mobilidade. A rede é descentralizada e, portanto, a organização e a entrega de mensagens devem ser executadas pelos próprios nós, que compartilham seus recursos [6].

Sua primeira aplicação foi em campo de batalha, onde redes estruturadas podem não estar disponíveis, dificultando a realização de operações que requeiram mobilidade [7]. Portanto, a utilização de uma rede *ad hoc,* na maioria dos casos, está associada a cenários onde não há uma infraestrutura de rede previamente instalada ou a mesma deixou de existir, como situação de desastres, incluindo furacões, terremotos, incêndios, inundações, guerras e atentados terroristas.

Outro aspecto a se considerar em tal ambiente é a maneira de como os dispositivos irão comunicar-se através da rede. Os protocolos *multicast-gossip* surgiram como uma técnica para implementar serviços altamente escaláveis e robustos, como a disseminação e agregação de informações, tornando-se uma abordagem interessante para MANETs, em que os nós perdem a conexão com frequência devido ao alcance e qualidade do sinal [8].

A comunicação através desses protocolos, devido à redundância, garante alta probabilidade de que todos os nós envolvidos na comunicação receberam as mensagens, de modo que uma informação se propague rapidamente em pouco tempo [9].

1.1 Motivação

Esta pesquisa se iniciou a partir de trabalhos sobre protocolos epidêmicos em redes sem fio, com interesse em determinar se tais protocolos seriam eficientes em cenários bem determinados, contrapondo o seu uso ao emprego de técnicas consagradas como *broadcast*[1]. Esta questão foi levantada a partir de [6], um estudo que analisou o protocolo *Network-Friendly Epidemic Multicast* (NEEM) na perspectiva de quantificação de falhas de comunicação. No estudo, foram cobertos vários cenários em redes móveis *ad hoc*, conhecidas como MANETs, incluindo velocidades de deslocamento de 2 a 50 Km/h e injeção de falhas. As análises foram feitas por modelagem estatística, simulação computacional e experimentos de campo, tendo como referência uma aplicação *peer-to-peer*[2] de envio de mensagens simples sobre o *Internet Control Message Protocol* (ICMP), mostrando resultados bastante satisfatórios nas métricas de taxa de entrega de mensagens, cobertura da rede, mensagens duplicadas e latência.

Além das sugestões de trabalhos futuros apresentadas naquele estudo, outras questões de interesse surgiram. Uma delas é relativa aos cenários investigados, pois não parece existir uma única aplicação que possa ser útil em condições estacionárias ou de baixa velocidade e, ao mesmo tempo, em situações em que veículos se cruzam em alta velocidade.

---

[1]Transmissão de mensagens para todos os receptores simultaneamente.

[2]Sistemas sem uma infraestrutura central.

No primeiro caso, tem-se, por exemplo, pessoas reunidas em um local ou dispersas numa área relativamente restrita, como ilustra a Figura 1 (casa, escola, centro de compras, terminal rodoviário, aeroporto, etc.), cenários em que as velocidades máximas são da ordem de 5 Km/h, as conexões entre os nós da rede são de longa duração e há também a expectativa de que os nós saiam e retornem à área de cobertura da rede *ad hoc* em intervalos limitados a minutos ou horas, quando a retenção de mensagens para posterior envio faz sentido. Neste caso, são exemplos razoáveis os jogos compartilhados, interações em redes sociais, serviços de localização, etc.

Figura 1- Ambiente MANET com baixa mobilidade

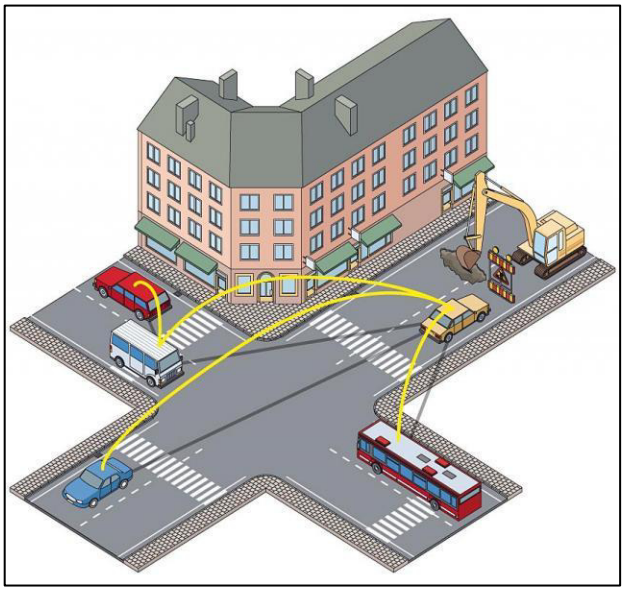
Fonte: Adaptada de ilustração livre obtida em banco de imagens[3].

No segundo caso, os dispositivos estão embarcados em algum tipo de veículo, as intersecções entre áreas de coberturas duram poucos segundos e não há expectativa de reencontro dos nós, como ilustra a Figura 2, o que torna inviável e injustificável qualquer tipo de armazenamento de mensagens para entrega futura. Aparentam beneficiar-se desta condição as aplicações voltadas ao estabelecimento de conexões de rede, de ampliação de cobertura e de atualização de dados compartilhados, como as condições de tráfego numa rodovia, por exemplo.

[3]Disponível em https://www.kisspng.com/png-meeting-desk-illustration-business-people-meeting-136417

Assim, como não parece haver aplicações reais úteis em cenários tão diversos, é válido indagar se as conclusões obtidas são precisas considerando um ambiente real.

Figura 2- Ambiente MANET com alta mobilidade



Fonte: Retirada de [10].

Ainda naquele estudo, a aplicação que gera a carga de trabalho para o ambiente de teste faz o envio de 30 mensagens simples de um único nó para a rede, sem réplica. Muito embora seja uma configuração experimental, este processo não se assemelha a uma aplicação real e, portanto, uma análise do comportamento da rede sob este experimento pode chegar a conclusões irreprodutíveis em um ambiente de produção.

Outro ponto interessante é o protocolo sobre o qual a aplicação foi construída, já que o ICMP é um protocolo de controle, não de transporte ou de aplicação. Por definição, um protocolo de transporte é aquele que estabelece efetivamente a troca de informações fim-a-fim, como o *Transmission Control Protocol* (TCP) e o *User Datagram Protocol* (UDP) [2]. Um protocolo de aplicação, por sua vez, é usado para dar suporte à construção de aplicações, muitas vezes clientes de um determinado serviço, possibilitando o desenvolvimento de distintos programas para o mesmo fim, a exemplo dos protocolos de correio eletrônico que fornecem suporte a diversos leitores de *e-mail*. A dúvida que surge é se o uso do ICMP não teria comprometido as conclusões do experimento.

1.2 Objetivo

O principal objetivo desta pesquisa foi avaliar o desempenho da comunicação *multicast-gossip* através do protocolo epidêmico NEEM, contrapondo com a técnica *broadcast*, em um cenário bem determinado, com uma aplicação realista construída sobre protocolos adequados, considerando o ambiente MANET.

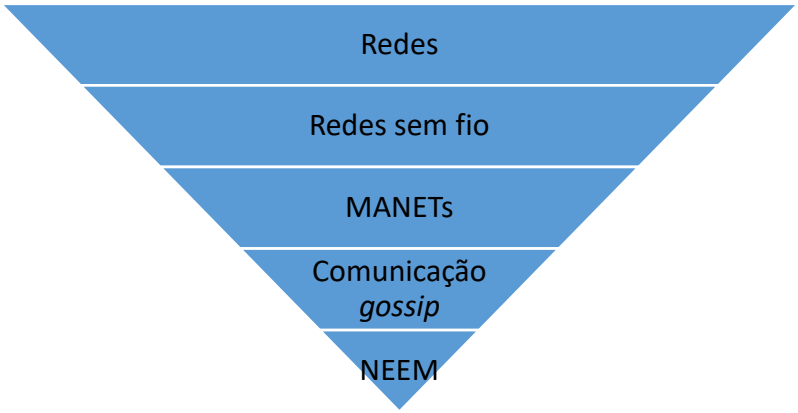
1.3 Proposta

Eliminando-se a situação estacionária, escolheu-se a velocidade de mobilidade no limite inferior dos testes de [6], da ordem de 2 Km/h e buscou-se um cenário real em que uma aplicação pudesse ser de interesse dos usuários de uma MANET nesta condição. Verificou-se que as MANETs tiveram sua origem ligada a operações militares, em campos onde a infraestrutura cabeada não seria disponível ou estaria muito danificada. Consequentemente, o seu emprego em situações de grandes desastres foi um passo lógico para estas redes. O atendimento a catástrofes é, por si, uma área na qual esforços são aplicados para tornar as operações de resgate mais eficientes. Um grande desafio nessas situações é a comunicação, principalmente quando estão envolvidas mais de uma organização nas atividades de socorro [11].

O primeiro desafio tecnológico após a ocorrência de um desastre é estabelecer a comunicação para os primeiros socorristas. Assim, as redes móveis sem fio formadas nestes ambientes podem melhorar a coordenação das equipes de resgate [12]. Um fator que favorece a montagem de redes de emergência é a popularização de dispositivos móveis como notebooks, PDAs (*Personal digital assistants*) e celulares com mais recursos disponíveis, que podem ser utilizados por agentes de equipes que atendem estas situações [13].

Portanto, uma aplicação que oferecesse suporte a equipes de resposta a emergências seria ideal para a pesquisa aqui pretendida, pois deixa a MANET em seu *habitat* de origem, implica na comunicação independente de infraestrutura física pré-existente, restringe a mobilidade a velocidades inferiores a 6 Km/h (pessoa caminhando rapidamente), apresenta-se realista e com utilidade razoável. A Figura 3 apresenta a estrutura proposta de estudo deste trabalho.

Figura 3- Estrutura do ambiente



Fonte: Produção do próprio autor.

Ao se ter acesso aos códigos usados na metodologia de [6] para as simulações sobre o programa NS3 (*Network Simulator* 3), percebeu-se que há discrepâncias, por vezes significativas, entre o que o texto indica e o que a implementação realmente é capaz de fazer. A título de exemplo, o trabalho indica um mecanismo de descoberta de vizinhança baseado no *broadcast* de mensagens ICMP. No entanto, o conjunto de nós da rede é fixado em código como uma faixa de endereços IP que cobre completamente o número de nós simulados, sendo vazio o código da função que deveria executar o procedimento de descoberta. Assim, assumiu-se que não seria viável utilizar a mesma metodologia daquele trabalho.

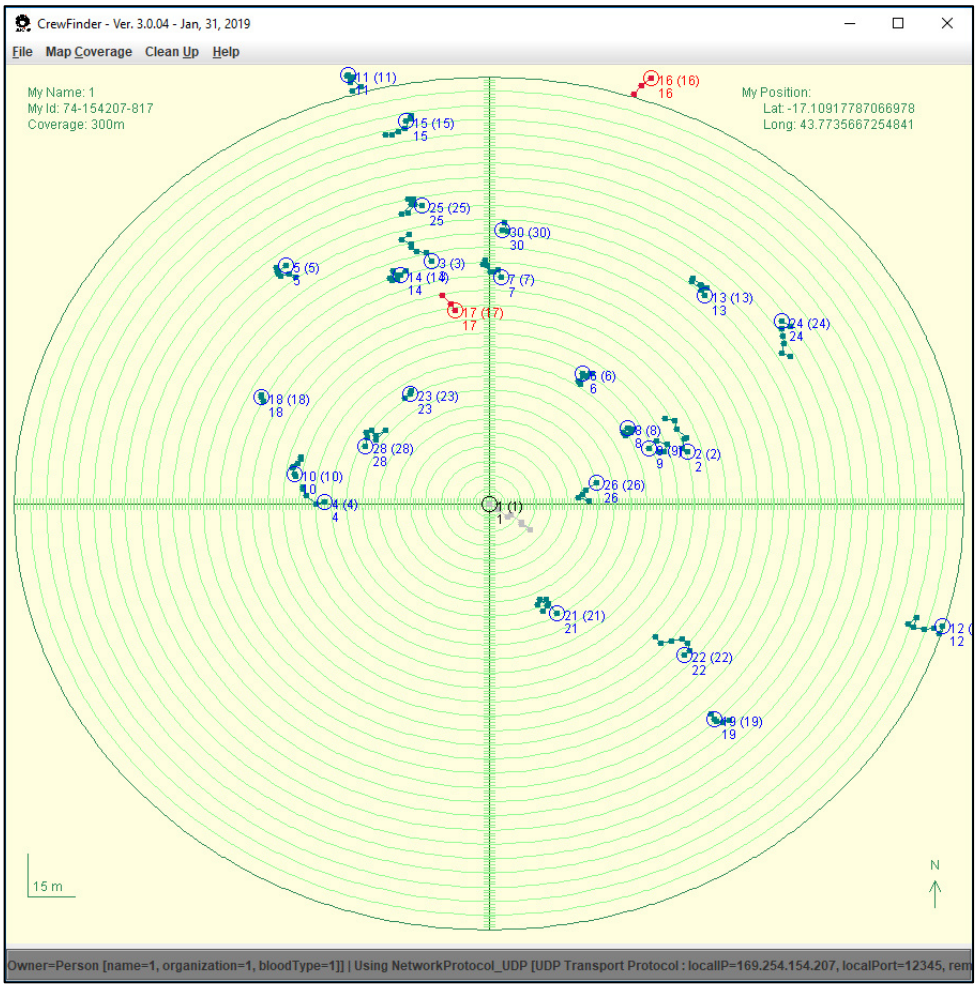
Então, foi projetada e desenvolvida para este trabalho uma aplicação em Java[4], denominada CrewFinder, cujo propósito é manter um mapa vetorizado da posição de cada membro de uma equipe de resposta a emergência. A CrewFinder foi inicialmente concebida como uma aplicação de estudos para permitir a geração de tráfego e sua medição, com vistas a responder à questão de eficiência dos protocolos epidêmicos *multicast* sobre a técnicas convencionais de *broadcast,* além do potencial de vir a ser útil para equipes trabalhando em ambientes de desastres. Sua arquitetura foi projetada em camadas de tal forma que uma interface Java define os métodos de envio e recepção de mensagens, tornando as funcionalidades da aplicação

[4]Linguagem de programação multiplataforma orientada a objetos.

independentes da interação com a rede, exatamente para que o mecanismo de comunicação possa ser alterado livremente nos experimentos.

A ideia geral da aplicação é que cada usuário, em seu dispositivo, apareça sempre no centro do mapa, mostrando detalhes como o nome, tipo sanguíneo, IP, cobertura e posição atual. Dentro do raio de alcance de visualização (que pode ser alterado no menu *Map Coverage*) são exibidas as posições atuais e vetorizadas dos companheiros de equipe. A aplicação em funcionamento durante um experimento, sob a perspectiva de visualização padrão (300 metros) pode ser vista na Figura 4.

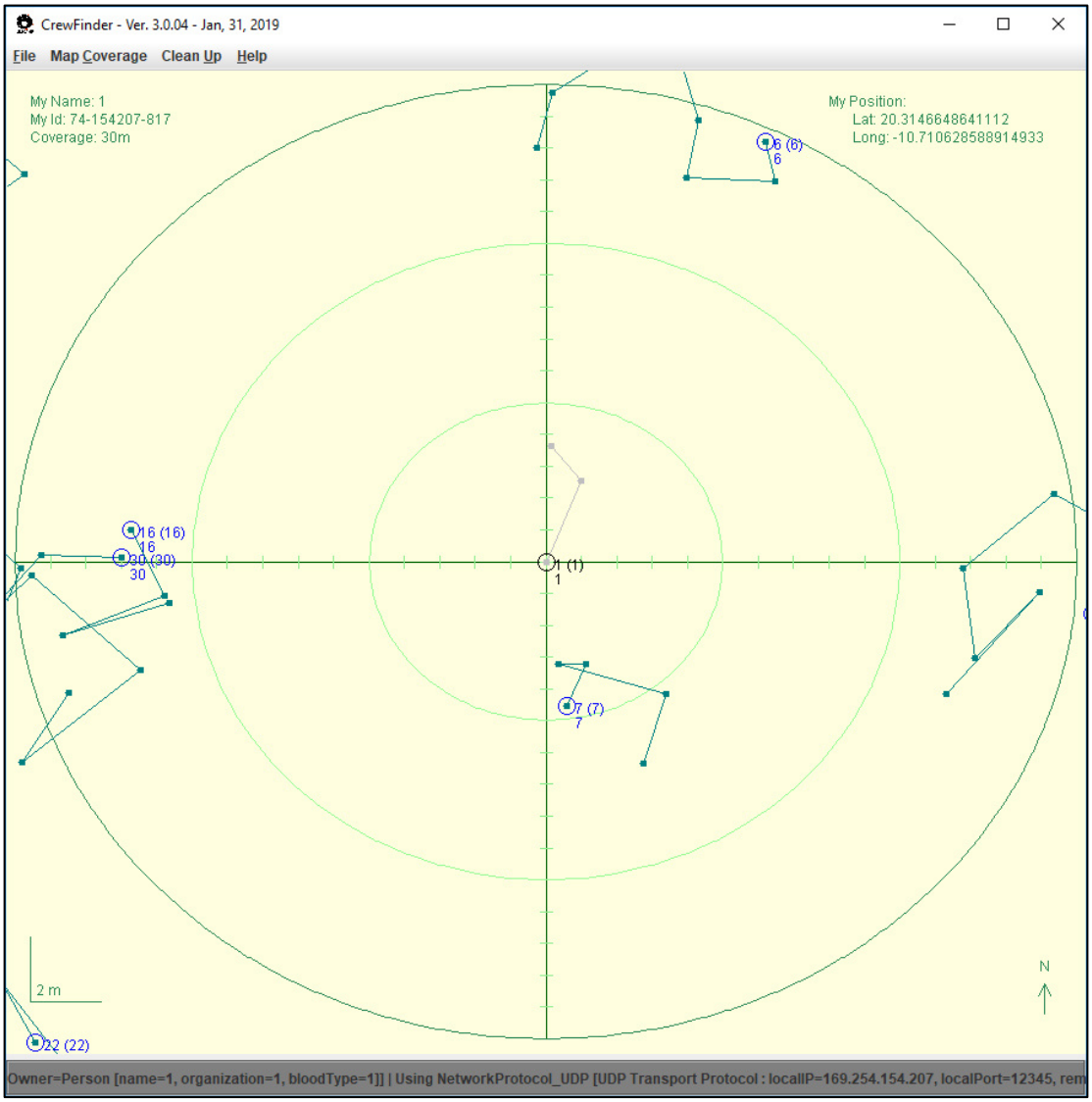
Figura 4- Interface da CrewFinder com visualização padrão



Fonte: Produção do próprio autor.

A maior ampliação que pode ser selecionada é de 30 metros, como exemplo na Figura 5. Nesta perspectiva, são vistas apenas as estações que estão bem próximas.

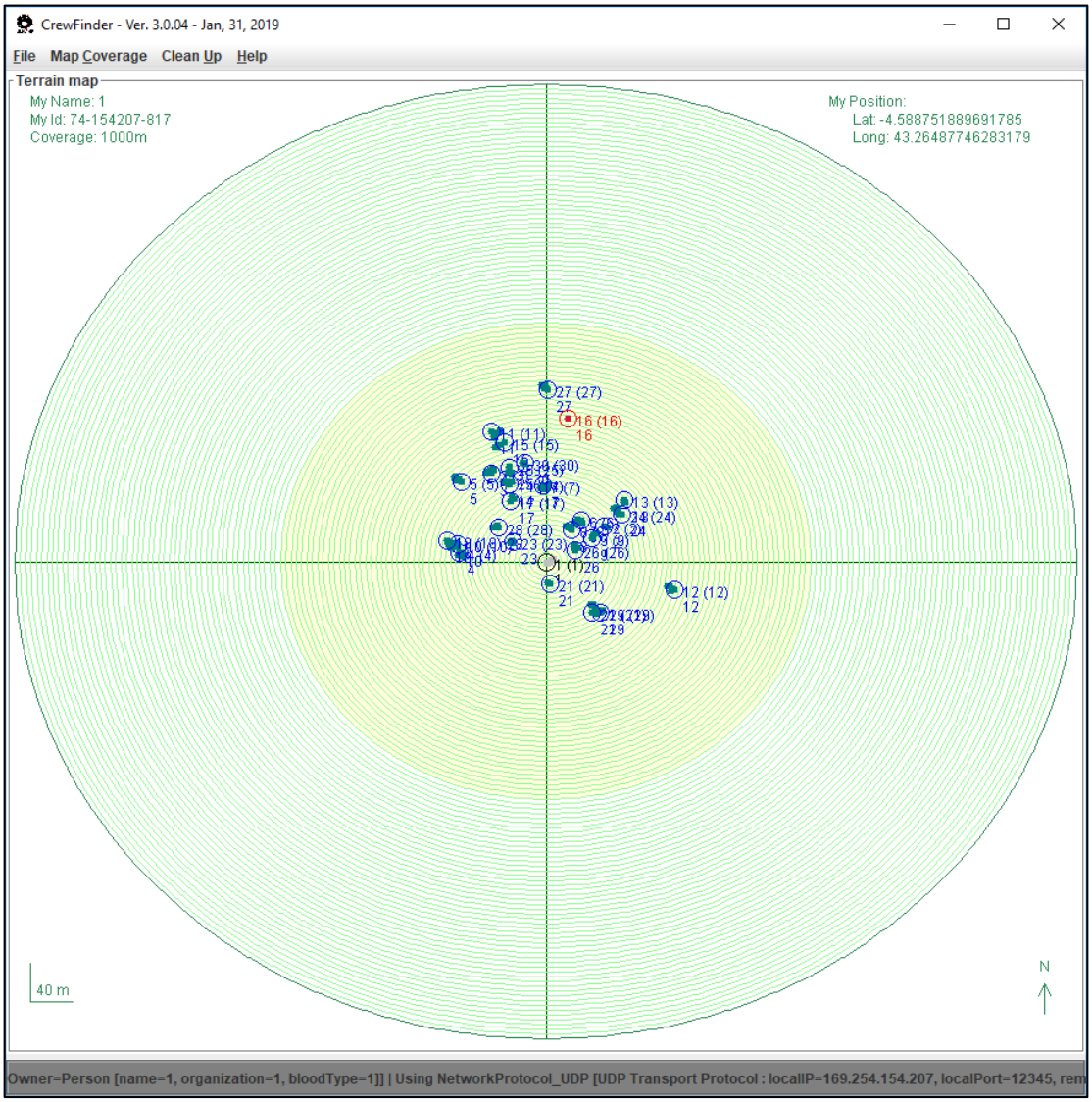
Figura 5- Interface da CrewFinder com ampliação máxima



Fonte: Produção do próprio autor.

Já a ampliação mínima que pode ser selecionada é de 1000 metros, como exemplo na Figura 6. Desse modo, mesmo as estações mais distantes também podem ser vistas.

Figura 6 - Interface da CrewFinder com ampliação mínima



Fonte: Produção do próprio autor.

O desenvolvimento da aplicação CrewFinder proporcionou o diferencial em permitir que todo estudo e experimentos deste trabalho fossem realizados em equipamentos físicos, executando uma aplicação real e comunicando-se através de uma rede MANET legítima, descartando a utilização de qualquer simulador.

O único mecanismo simulado é a movimentação dos usuários pelo terreno, descrito com mais detalhes na seção 3.2.1. Para tornar esta simulação um processo real, uma reformulação total seria necessária em termos de estrutura, recursos e metodologia que levaria este trabalho a um outro patamar e o tornaria inviável de ser desenvolvido.

1.4 Estrutura do trabalho

Os próximos capítulos deste texto apresentam, pela ordem, os seguintes conteúdos: no Capítulo 2, os fundamentos teóricos, com uma contextualização sobre as redes sem fio, com ênfase nas MANETs, uma revisão conceitual crítica sobre os principais protocolos epidêmicos e uma descrição detalhada do funcionamento e dos parâmetros dos protocolos epidêmicos em geral, além de uma abordagem do NEEM em particular, correlacionadas à implementação da NeEM *library*; no Capítulo 3, toda metodologia usada em cada etapa da pesquisa, inclusive a aplicação CrewFinder, indicando sua arquitetura, com destaque para a interface flexível de comunicação de rede; no Capítulo 4, os resultados obtidos nos experimentos e a discussão sobre o trabalho desenvolvido; e no Capítulo 5, as conclusões.

2. Fundamentos teóricos

Neste capítulo, são apresentados fundamentos teóricos sobre os principais conceitos abordados neste trabalho, como redes sem fio e protocolos epidêmicos.

2.1 Redes sem fio e redes *ad hoc*

Amplamente encontradas nos mais diversos tipos de estabelecimentos, as redes sem fio são hoje uma das tecnologias de acesso à Internet mais importantes [14]. As *Wireless LANs (*WLANs – LANs sem fio*),* popularmente conhecidas como *Wi-Fi*, foram especificadas pelo padrão IEEE 802.11 [8] em 1997, que define os formatos e as estruturas dos sinais de curto alcance que fornecem o serviço *Wi-Fi* [4], posteriormente originando outros padrões como 802.11b, 802.11a, e os mais utilizados atualmente, 802.11g e 802.11n, além do 802.11p, destinado a redes veiculares *ad hoc* (VANETs), conforme ilustrado na Tabela 1.

Tabela 1- Resumo dos padrões IEEE 802.11

Padrão	Frequência de Rádio	Taxa de Dados
802.11b	2.4 GHz	Até 11 Mbps
802.11a	5 GHz	Até 54 Mbps
802.11g	2.4 GHz	Até 54 Mbps
802.11n	2.4 GHz	Até 300 Mbps
802.11p	5.9 GHz	Até 27 Mbps

Fonte: Adaptada de [15].

Tal especificação, de modo geral, define que a base (BSS – *Basic Service Set*) de uma WLAN pode ser formada por estações fixas ou móveis. Ainda nesse sentido, tais redes podem ser classificadas em duas categorias, de acordo com sua organização. Quando existe uma estação-base, conhecida como AP (*Access Point*), responsável por fazer a interconexão entre os dispositivos e a Internet, são ditas redes de infraestrutura. Em tese, são as amplamente encontradas em residências. Na outra categoria, encontram-se as redes com a ausência de uma estação-base central, de forma isolada e independente, conhecidas como redes *ad hoc*, capazes de se localizar e conectar entre si [2]. Um exemplo prático é o de pessoas portando dispositivos móveis, em um ambiente próximo como uma sala de reunião, com a necessidade de se comunicar e não dispondo de infraestrutura de rede. Neste caso, pode ser formada uma rede *ad hoc*.

Com o crescente uso de dispositivos móveis, impulsionado pela popularização da comunicação sem fio, as MANETs vêm ganhando importância com o aumento de diversas aplicações [16]. Sua flexibilidade permite que sejam utilizadas em missão de resgate, desastres naturais, usos educacionais, além de diversas outras [17].

Porém, é importante fazer a distinção entre sem fio e móvel, já que existem muitos casos nos quais os dispositivos são sem fio, mas não necessariamente móveis, como pessoas utilizando notebooks em suas mesas de trabalho em um escritório, por exemplo. Quando a característica de mobilidade se faz presente, as redes móveis *ad hoc* são conhecidas como MANETs *(Mobile ad hoc networks),* as quais esta pesquisa considerou para estudo.

As principais características das MANETs são [18]:

a) A topologia da rede é altamente dinâmica, tornando difícil a prevenção de mudanças;
b) MANETs são baseadas em links *wireless,* consequentemente possuem maiores limitações comparadas a redes cabeadas;
c) A segurança física é limitada devido ao meio de transmissão*;*
d) MANETs sofrem, na maioria dos casos, com altas taxas de perdas *(loss rates)* e atrasos (*delays/jitter*), consequentes da transmissão *wireless*.

Outro aspecto a se considerar neste ambiente é o consumo de energia. Como espera-se que os usuários desta rede estejam portando um dispositivo que depende de bateria, a economia desta é um importante critério a se considerar no projeto de um sistema voltado a este cenário.

As redes *ad hoc* estão despertando um grande interesse com a contínua proliferação de equipamentos portáteis que podem se comunicar. O desenvolvimento de protocolos para essas redes é desafiador e constitui o assunto de trabalhos em diferentes frentes [14].

2.2 Revisão dos protocolos epidêmicos

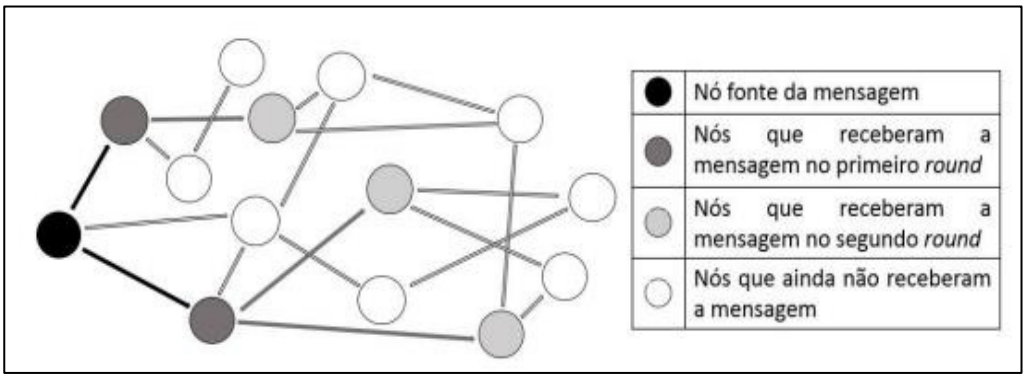
O primeiro conceito de algoritmo epidêmico apareceu em 1987, em [19], resultado de uma pesquisa realizada nos laboratórios da *Xerox Corporate Internet,* que buscava uma solução para melhorar a consistência de dados entre diferentes servidores distribuídos. Esta pesquisa foi realizada sobre um algoritmo existente na época *(Direct Mailing),* que consistia em fazer com que o servidor que gerasse uma atualização enviasse uma mensagem informando todos os outros. Essa abordagem, apesar de ser eficiente por gerar apenas o número necessário de mensagens, não tinha confiabilidade, visto que as mensagens poderiam ser perdidas por falhas nos servidores de e-mail, além de que cada servidor precisava ter conhecimento completo da rede para poder enviar uma mensagem a todos os outros.

Os pesquisadores, então, partiram do princípio da disseminação de um rumor como um possível mecanismo de solução para as limitações que tinham: num determinado conjunto de pessoas, uma dentre elas inicia um rumor, seleciona um vizinho de forma aleatória e compartilha este rumor, este é compartilhado com outro vizinho aleatório e assim sucessivamente, continuando o processo de disseminação da "fofoca" (por isso também são conhecidos como protocolos *multicast-gossip*). Outra analogia é uma epidemia se espalhando pela população, daí o termo "protocolo/comunicação epidêmica".

Em teoria, essa abordagem resolveria os problemas do algoritmo existente, já que os servidores não mais precisariam conhecer a estrutura completa da rede, além de garantir maior disseminação da informação, pois eventualmente um servidor com problemas não impediria outros de receberem a mensagem.

Dessa maneira, os protocolos *multicast-gossip* surgiram como uma técnica para implementar serviços escaláveis e robustos, como a disseminação e agregação de informações, cuja aplicação em MANETs torna-se interessante devido limitações conhecidas desta arquitetura de rede [8]. A Figura 7 ilustra um exemplo de como ocorre a comunicação *gossip*.

Figura 7- Disseminação de uma mensagem *gossip*



Fonte: Adaptada de [9].

Da publicação inicial do conceito até hoje, muitos estudos de otimizações foram feitos sobre a abordagem, resultando na criação de diversos "protocolos" epidêmicos com características diferentes, apesar do mesmo princípio *(gossip).* A seguir, são apresentados alguns:

NEEM: *NEtwork-friendly Epidemic Multicast,* sua principal característica é combinar o uso do TCP para controle de largura de banda disponível e técnicas de gerenciamento de *buffer* para descarte de mensagens em excesso, visando garantir que não haja sobrecarga da rede [20].

HyParView: *Hybrid Partial View*, também utiliza TCP para aumentar a confiabilidade e tratamento de falhas. Esse protocolo tem como característica manter duas listas de vizinhos, os ativos e os passivos, de modo que se um vizinho falhar, um passivo é promovido, mantendo sempre cada nó conectado com um vizinho ativo. É uma derivação do protocolo NEEM [21].

HEAP: *HEterogeneity-Aware Gossip Protocol* tem como objetivo aumentar a eficiência da disseminação das mensagens alterando dinamicamente o parâmetro *fanout*[5] dos nós. Nós mais eficientes (com maior quantidade de banda disponível) tem seu *fanout* aumentado,

[5] O parâmetro *fanout* é abordado na seção 2.2.2

enquanto os menos eficientes são diminuídos na mesma proporção. Dessa forma, a heterogeneidade (a diferença entre capacidade de banda disponível em cada nó) da rede tende a ser respeitada [22].

CREW: Usa conexões TCP para estimar a largura de banda disponível, otimizando assim o parâmetro *fanout* da comunicação *gossip*. A ênfase deste é otimizar a latência, sendo uma característica chave manter um *cache*[6] de conexões com pares descobertos usando um protocolo de caminhada aleatória, para evitar a latência de abrir uma conexão TCP quando um novo nó é inserido [23].

A escolha do NEEM como objetivo deste trabalho está correlacionada com pesquisas que foram feitas na Faculdade de Tecnologia no momento do início do mestrado, se mostrando um candidato viável para análises comparativas e amplamente citado na literatura científica. Também se destacam aspectos do NEEM como sua proposta em ser amigável a rede, uso do TCP e possibilidades de configuração.

De acordo com [9], que fez um estudo comparativo entre os protocolos epidêmicos mais conhecidos, apesar de terem sido utilizados com objetivos e áreas de atuação diferentes, mantiveram como foco a rápida propagação de informação, que é a finalidade principal do protocolo de comunicação baseado em *gossip*.

Na seção seguinte, são apresentados de forma mais detalhada o funcionamento e os parâmetros essenciais da comunicação *gossip.*

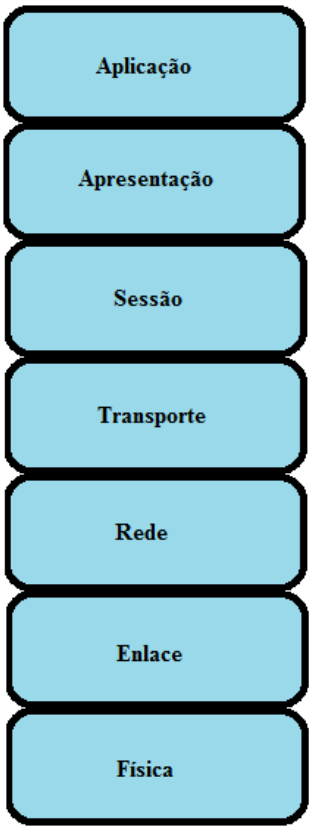
2.2.1 Arquitetura e pré-requisitos

O conceito de protocolo, amplamente difundido na comunidade de Redes de Computadores, define o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento de uma mensagem. Cada protocolo pertence a uma camada, a qual provê seu serviço executando certas ações dentro dela e utilizando os serviços da camada diretamente abaixo dela [14]. Eles são

[6]Memória temporária de alto desempenho.

para a comunicação por computador o que linguagens de programação são para a computação [24]. No final dos anos 1970, a Organização Internacional para Padronização (ISO - *International Organization for Standardization*) propôs que as redes de computadores fossem organizadas em 7 camadas, denominadas modelo de Interconexão de Sistemas Abertos (OSI – *Open System Interconnection*) [14], que pode ser visto na Figura 8.

Figura 8- Modelo de camada ISO/OSI



Fonte: Produção do próprio autor.

Duas entidades não podem simplesmente enviar fluxos de *bits* uma para a outra e esperar que sejam compreendidas; para que ocorra a comunicação, elas devem concordar com um protocolo, que é um conjunto de regras que controlam as comunicações de dados, definindo o que é comunicado, como isso é comunicado e quando deve ser comunicado [2].

Ainda segundo [2], os elementos-chave de um protocolo de rede são sintaxe, semântica e *timing*. Sintaxe refere-se à estrutura ou o formato dos dados, significando a ordem na

qual eles são apresentados. Semântica se refere ao significado de cada seção de *bits*, e *timing* refere-se a duas características: quando os dados devem ser enviados e com que rapidez eles podem ser enviados.

Uma dificuldade deste trabalho foi, justamente, classificar o NEEM como protocolo em relação a alguma arquitetura de rede como a ISO/OSI, apresentada na Figura 8, ou a TCP/IP. Não parece se tratar de um protocolo de transporte, pois depende do TCP, mas também não se encontrou uma evidência clara de que pudesse ser bem categorizado como um protocolo de aplicação. Um protocolo de aplicação define como processos de uma aplicação, que funcionam em sistemas finais diferentes, passam mensagens entre si. Eles definem [14]:

a) Os tipos de mensagens trocadas, por exemplo, de requisição e de resposta;
b) A sintaxe dos vários tipos de mensagens, tais como os campos de mensagens e como os campos são delineados;
c) A semântica dos campos, isto é, o significado da informação nos campos;
d) Regras para determinar quando e como um processo envia e responde as mensagens.

Na literatura científica, notou-se grande dificuldade em encontrar tais características tanto no NEEM quanto nos demais protocolos de fofoca. Houve diversas tentativas de definir formalmente os protocolos *gossip,* mas não existe uma definição padrão [25]. De acordo com [25] e [26], as seguintes propriedades devem ser consideradas:

a) A seleção dos nós vizinhos deve ser aleatória;
b) Apenas informações locais estão disponíveis em todos os nós;
c) A comunicação é periódica;
d) A transmissão e a capacidade de processamento são limitadas;
e) Todos os nós executam o mesmo protocolo.

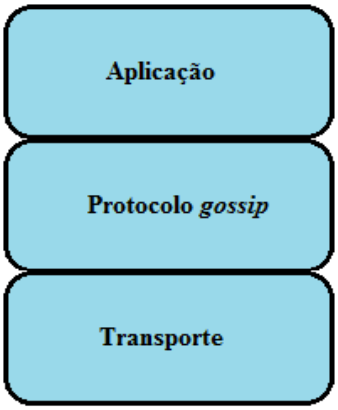
Porém, de acordo com [27], para ser considerado *gossip* um protocolo deve satisfazer as seguintes condições:

a) Envolve comunicação periódica entre os pares de nós;
b) A informação trocada é de tamanho reduzido e limitado;

c) Cada interação entre nós resulta na alteração do estado de um ou ambos;
d) Comunicação confiável não é um requisito;
e) A frequência das interações é baixa comparada às latências de mensagens típicas, de modo que os custos do protocolo são insignificantes;
f) Existe alguma forma de aleatoriedade na escolha dos nós com quem interagir. Eles podem ser escolhidos apenas entre os vizinhos ou entre todos os nós.

Há, ainda, quem considere protocolos epidêmicos como um *middleware*[7], que geralmente é implementado entre a camada de aplicação e a camada de transporte [21]. A Figura 9 mostra um resumo de uma arquitetura baseada em *gossip.*

Figura 9- Arquitetura genérica do *gossip*



Fonte: Adaptada de [21].

Ainda em relação ao NEEM, os autores citam [19] e [28] como fontes consultadas com referência a protocolos epidêmicos, mas nestes trabalhos os termos mais frequentes são algoritmos epidêmicos, processos epidêmicos, métodos epidêmicos e mecanismos de comunicação epidêmica e, em nenhum deles é definido o que seja um protocolo epidêmico. Aparentemente, ambos usam o termo protocolo em um sentido amplo, de forma nenhuma ligada aos

[7] Programa que fornece serviços para outros softwares além dos já oferecidos pelo sistema operacional.

modelos de arquiteturas de Redes de Computadores. Questionado sobre este dilema, José Orlando Pereira, principal autor do NEEM, afirmou que “Embora nunca me tenha preocupado muito com essa classificação, diria que é um protocolo de redes, ao nível da aplicação” [29].

Assim sendo, ao que parece, a terminologia “protocolo”, amplamente empregada na comunicação epidêmica, foi adotada pela comunidade que estuda e desenvolve tais tecnologias sem muitas preocupações com as definições formais e requisitos que caracterizam de fato um protocolo de rede, apesar deste trabalho considerar mais adequado o uso do termo “protocolo” no sentido de “processos/procedimentos” de comunicação, de maneira não ligada a protocolos de rede. Na realidade, o termo em si envolve uma lista bastante extensa de significados e pode ser aplicado em diferentes contextos. Não é do objetivo deste trabalho discutir a terminologia em si.

Na seção 2.2.2 são exibidos os principais conceitos a respeito do funcionamento dos mecanismos epidêmicos.

2.2.2 Funcionamento e parametrização

Na comunicação epidêmica, tem-se 3 estados possíveis para os nós, que foram nomeados com base na epidemiologia. São eles:

a) Estado suscetível: nó que ainda não recebeu o rumor;
b) Estado infectado: nó que recebeu o rumor e o está compartilhando;
c) Estado removido: nó que recebeu o rumor, porém não o dissemina mais.

Para que se tenha controle sobre tais estados, como decidir quando um nó passa do estado infectado para removido ou então o momento de parar de replicar um rumor, são necessários certos parâmetros de controle. Inicialmente, tais parâmetros foram apresentados no desenvolvimento do protocolo *pbcast* [30] em 1999, também descrito em [31] em 2004, conforme segue:

a) *Fanout*: número de vizinhos selecionados, geralmente de maneira aleatória, para receber a mensagem enviada pelo transmissor. Alto valor de *fanout* garante uma

elevada probabilidade de entrega, porém gera um aumento de tráfego na rede devido à redundância;

b) *Hops/Rounds/*TTL (*Time to Live*): Indica o número máximo de vezes (ou saltos) que uma mensagem será retransmitida pelos nós.

Além destes 2 parâmetros de controle, também são necessárias estratégias de como se dará a disseminação das mensagens pelo *multicast-gossip.* As principais são mostradas abaixo:

a) *Earger Push*: Os nós enviam mensagens para vizinhos selecionados aleatoriamente, logo que as recebem pela primeira vez;
b) *Pull*: Os nós consultam seus vizinhos de forma aleatória para obter informações sobre as mensagens recebidas recentemente e solicitam a mensagem quando percebem que ainda não a receberam;
c) *Lazy Push*: Ao receber uma mensagem pela primeira vez, o nó verifica apenas o identificador e não a carga completa. Se os nós recebem um identificador de uma mensagem que não tenham recebido, fazem uma solicitação de recebimento da mensagem completa ao outro nó.
d) *Hybrid*: A comunicação epidêmica da mensagem é feita em duas fases distintas. A primeira fase utiliza *push* para divulgar uma mensagem com melhor esforço e a segunda utiliza *pull*, a fim de recuperar omissões produzidas na primeira fase.

Tanto os parâmetros como as estratégias podem ser implementadas de maneiras diferentes de acordo com cada protocolo e tem papel fundamental no desempenho do processo *multicast-gossip* [6].

2.3 Network Friendly Epidemic Multicast - NEEM

Nesta seção são explicadas as características do NEEM e, em 2.4, é apresentada sua implementação existente na linguagem Java, denominada NeEM *library*, a qual possibilita construir aplicações que se comunicam sobre o mesmo.

O *Network Friendly Epidemic Multicast* - NEEM foi proposto em [20], em 2003, apresentando seu diferencial em utilizar conexões TCP/IP, visto que protocolos *multicast*, por

se tratar de um tipo de comunicação que não é ponto a ponto, geralmente são baseados em UDP. Os autores justificam que o uso do TCP traz benefícios interessantes, sendo o principal deles prover controle de congestionamento em períodos de sobrecarga na rede, inclusive utilizando técnicas de gerenciamento de *buffers* para armazenamento de mensagens, evitando o descarte de imediato como ocorre com o UDP, além do fato de que tráfego TCP/IP geralmente tem prioridade em *Internet Service Providers* (ISPs). Assim sendo, o propósito principal deste protocolo, como sugere o nome, é ser amigável a rede.

Examinando-se [20] em detalhes, vê-se que o NEEM foi baseado no *Pbcast Protocol*, descrito em [30] como tendo propriedades que são úteis para a construção de protocolos ao estilo de votação, remanescentes dos protocolos de *quorum* dos bancos de dados. Adicionalmente, o *Pbcast* tem as seguintes características: no máximo uma entrega, garantia de integridade, entrega ordenada de mensagens e confiança probabilística (termo usado pelos autores para se referir à consequência da distribuição binomial de entregas). O NEEM combina o *Pbcast* com dois mecanismos complementares: a) um controle de congestão fim-a-fim amigável para o TCP, permitindo o uso seguro da largura de banda disponível; b) uma técnica de gerenciamento de *buffers*[8], de modo que as mensagens sejam retidas ao invés de diretamente injetadas na rede, que combina diferentes políticas de seleção para descarte em situação de transbordamento (congestionamento).

Os autores descrevem a organização do protocolo como a combinação de três camadas complementares entre si: a primeira, no topo, é a responsável por fazer a comunicação epidêmica propriamente dita, baseada no *Pbcast protocol.* No processo, as mensagens são inicialmente marcadas com o número máximo de *rounds* (também conhecidos por *hops* ou TTL) e distribuídas para diferentes nós (*fanout*) aleatoriamente, sendo a eficiência oferecida pelo protocolo dependente da configuração apropriada destes parâmetros. Ao receber uma mensagem, o número de *rounds* restantes é diminuído até chegar a zero, quando é descartada. As outras duas camadas tratam do gerenciamento de *buffer* e configuração da camada de transporte através do TCP.

[8]Memória para armazenamento de dados temporários.

Ainda em relação ao *buffer,* os autores apresentam a combinação de 3 diferentes políticas, por ordem de preferência, para descarte das mensagens em caso de sobrecarga:

a) Descarte randômico: ao chegarem novas mensagens, algumas são selecionadas aleatoriamente para descarte. Esta estratégia é interessante para ser utilizada apenas como segunda opção;
b) Descarte por idade: descarta a mensagem que foi disseminada mais vezes, visto que é provável que os outros nós já tenham recebido. A ideia é que a idade seja carregada na própria mensagem e cada nó ao retransmiti-la acrescente idade +1;
c) Descarte semântico: descarte de mensagens consideradas obsoletas pela aplicação. Para tanto, cada mensagem deve carregar um mapa de bits que, combinado com um identificador único entre as mensagens, tem-se que, "se o *i*-ésimo bit estiver ligado no *bitmap*[9] pertencente a mensagem *j*, o protocolo é informado que a mensagem com número de sequência *j – i* é considerada obsoleta [32].

Desta forma, os autores concluem dizendo que este é o primeiro protocolo epidêmico proposto que suporta conexões fim-a-fim (pelo uso do TCP) e seguro que pode ser utilizado na Internet, visto que os existentes na época haviam sido projetados apenas para redes locais, além de contar com uma sofisticada técnica de gerenciamento de *buffer.*

O NEEM foi descrito pela primeira vez em [20], onde se afirma que os protocolos epidêmicos, também chamados de probabilísticos ou baseados em fofoca (*gossip-based*), são uma proposta atrativa para atender aos desafios de desempenho e escalabilidade de *multicast* confiável. Como vantagens adicionais, os autores citam que protocolos epidêmicos *multicast* suportam vazão estável mesmo para grupos grandes com nós sobrecarregados, já que a carga requerida para garantir a confiabilidade é igualmente espalhada entre todos os membros do grupo e um único nó sobrecarregado não pode bloquear os emissores, como acontece em outros protocolos.

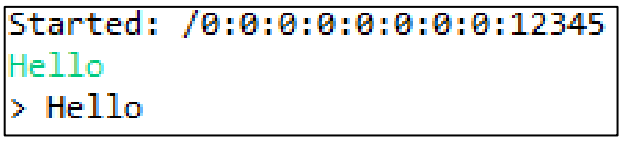
[9]Matriz de bits.

2.4 NeEM *library*

Durante o desenvolvimento deste trabalho, foi encontrada em [33] uma implementação do NEEM na linguagem Java (NeEM *library*), baseada nas interfaces NIO para manter múltiplas conexões de rede sem a sobrecarga dos *threads*, tendo como meta ser pequena, autônoma e prática. Isto abriu novas possibilidades para esta pesquisa, já que um programa em execução gerando tráfego de rede pode ser medido e não somente simulado. A partir daí, ganhou força a ideia de implementar a aplicação de testes, denominada CrewFinder.

A NeEM *library* fornece uma implementação do protocolo *multicast* epidêmico NEEM, permitindo a comunicação em rede através de conexões TCP/IP, o que possibilita a disseminação de mensagens para um grande conjunto de nós. A biblioteca fornece uma aplicação de exemplo, um *chat* de troca de texto simples, no qual as mensagens digitadas no console são enviadas para todos os nós do grupo com as técnicas de *gossip,* como pode ser visto na Figura 10. Podem ser conectados quantos forem necessários e a própria biblioteca faz o gerenciamento das conexões, garantindo que os parâmetros, como o *fanout*, sejam respeitados.

Figura 10- Aplicação chat em funcionamento



Fonte: Produção do próprio autor.

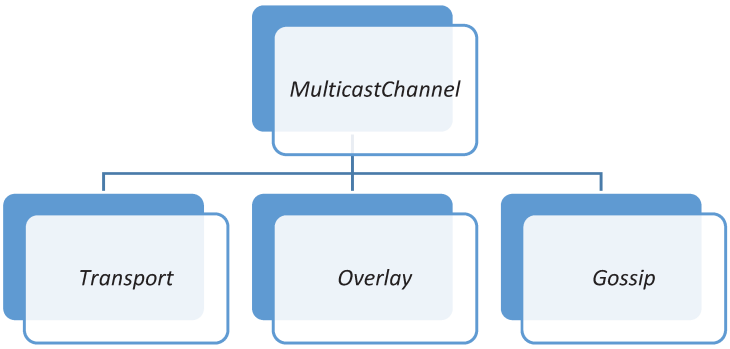
A seguir, apresenta-se de forma resumida o funcionamento das principais classes, baseado em [32], que fez um estudo aprofundado do funcionamento da biblioteca. A versão atual, 0.8, analisada neste trabalho, encontra-se disponível no repositório *GitHub*[10].

Para utilizar a biblioteca, deve-se primeiramente instanciar a classe de interface com o usuário *MulticastChannel,* fornecendo os parâmetros de endereço IP e porta de comunicação a serem usadas para abrir as conexões TCP/IP com os vizinhos. Os principais métodos

[10]https://github.com/jopereira/neem

desta são *MulticastChannel.read* e *MulticastChannel.write,* que periodicamente são executados para ler/escrever dados no canal. Ao se criar o objeto *MulticastChannel*, são geradas instâncias das classes *Transport, Overlay* e *Gossip,* como visto na Figura 11.

Figura 11- Principais classes da NeEM *library*



Fonte: Produção do próprio autor.

A classe *Transport* é responsável por simular a camada de rede, servindo de interface para envio/recebimento de pacotes para as demais. Ela possui os métodos necessários para manipulação das conexões (abrir/fechar, escrever/ler dados), ou seja, na existência de qualquer evento relacionado a uma conexão, ela notifica o objeto *Transport*. Quando um usuário em posse de um objeto *MulticastChannel* deseja escrever um dado no canal ou comunicar-se com os demais, cria-se uma tarefa no objeto *Transport,* que as organiza em uma fila para serem executadas no próximo nano-segundo sem tarefas a executar, mantendo um controle das requisições recebidas de outros objetos.

Para permitir que outros objetos possam "escutar" os dados trafegados, *DataListeners* são criados no objeto *Transport*, associados as suas respectivas portas de comunicação, da mesma forma que *ConnectionListeners* também podem ser criados possibilitando o monitoramento de eventos como conexões criadas ou destruídas, permitindo assim que o objeto *Overlay* seja informado de novos vizinhos ou exclusão dos existentes, por exemplo. Cabe ao próprio objeto *Transport* ativar o *DataListener* ou *ConnectionListener* adequado quando pacotes são recebidos ou conexões são abertas/fechadas.

A classe *Overlay* tem como principal função gerenciar o anúncio e criação da vizinhança de um nó. É nela que, por exemplo, que fica o parâmetro *fanout,* que limita a quantidade

vizinhos selecionados para envios de mensagens (valor 11 por padrão na versão utilizada), subtraindo essas responsabilidades da classe *Gossip*. Como a tarefa de criação e fechamento de conexões é da classe *Transport,* a classe *Overlay* se comunica de forma constante com a mesma.

Para gerenciar esse controle, um objeto *Overlay* utiliza 3 *DataListeners* criados em *Transport* chamados *ID*, *Shuffle* e *Join*. Sempre que uma conexão é criada, uma mensagem é enviada pelo canal *ID*, cujo nó receptor adiciona o emissor como um novo vizinho e envia uma solicitação pelo canal *Join* para seus já conhecidos vizinhos, pedindo que o novato seja apresentado em suas próprias vizinhanças. Ao receber a mensagem proveniente de *Join*, os receptores enviam uma mensagem de anúncio da presença do novo nó para seus respectivos vizinhos através do canal *Shuffle*, com o intuito de adicioná-lo ao maior número possível de listas de vizinhos. Contudo, para que um novo nó entre em uma lista existente, certos critérios devem ser respeitados:

a) O nó receptor não conter esse vizinho em sua lista;
b) O nó receptor ter uma quantidade de vizinhos menor do que *Overlay.fanout*.

Ainda que o critério B não seja respeitado, o novo nó tem uma chance de 50% de entrar na lista. Para manter o controle sobre o parâmetro *fanout*, também no momento da criação de uma nova conexão, algumas existentes são escolhidas aleatoriamente para serem fechadas. Além disso, a cada 10 segundos, são escolhidos dois nós aleatórios e apresentados um aos outros, através do canal *Shuffle.*

Por fim, o objeto da classe *Gossip* é responsável por fazer a disseminação dos dados aplicando as técnicas de "fofoca", que são fornecidos pelo usuário através da *MulticastChannel,* para os vizinhos gerenciados por um objeto da classe *Overlay*. Para controle da disseminação, a classe *Gossip* faz uso dos parâmetros *fanout* e TTL, explicados anteriormente. Na prática, para disseminação das mensagens, são criados dois *DataListeners* em *Transport:*

a) Um de dados, onde trafegam as mensagens;
b) Outro de controle, que trafegam anúncios de ACK (confirmação das estações que já receberam a mensagem) e NACK (estações que tem interesse em recebê-la).

Tais pacotes possuem três campos: um identificador único de 16 *bytes*, um campo de 1 *byte* para registrar a quantidade de nós que a mesma já passou (TTL), e o conteúdo, com tamanho variável – vazio quando se trata de um anúncio. Além disso, também são mantidas duas estruturas FIFO *(Fist-in e First-out)*, uma para controle de mensagens conhecidas *(cached)* e outra para controle de anúncios conhecidos *(queued),* de maneira que, se um nó recebe um anúncio de ACK e não possui a mensagem na fila *cached*, ele adiciona o anúncio e o remetente à fila de *queued*, para saber quem de seus vizinhos possui a mensagem que ele desconhece. Quando recebe um anúncio de NACK e a possui a mensagem na fila *cached,* ele repassa a mensagem completa. Os procedimentos que ocorrem no recebimento de uma mensagem por um nó são:

a) É verificado se o *hop* da mensagem é menor ou igual do que *Gossip.TTL,* caso negativo, encerra-se a tratativa sem disseminá-la;
b) Caso já exista em *cached,* é descartada. Caso contrário, é inserida na fila;
c) Se tratar de um anúncio já existente em *queued,* é descartado, caso contrário é inserido na fila;

Como o NEEM adota a maneira de disseminação híbrida, os nós selecionados para receberem a nova mensagem (*Gossip.fanout*) são divididos nos que receberão apenas o anúncio e os que receberão a mensagem completa.

Além das principais classes explicadas anteriormente, para fazer a integração com a CrewFinder foram necessárias utilizar certas classes de suporte, que foram organizadas em um pacote denominado “neemPackage”. Tais classes foram incorporadas na CrewFinder de maneira gradual, na medida que eram solicitadas como dependência por outras, com o intuito de trazer somente o que fosse necessário para o funcionamento dos mecanismos do NEEM. Uma breve descrição das classes de suporte que foram incorporadas na CrewFinder é exibida na Tabela 2*,* e foi baseada na documentação completa do software, que pode ser consultada no *SourceForce*[11].

[11] https://neem.sourceforce.net/api

Tabela 2- Classes de suporte da NeEM *library*

Classe	Descrição	Tipo
Acceptor	Monitora *sockets* que estão aceitando conexões. Derivado da classe Handler.	Classe
Addresses	Provê métodos para manipulação de endereços, organização IP e sequências de bytes.	Classe
AddressParser	Teve seu nome original alterado para "AdressParser" neste trabalho. Responsável pela interpretação e validação dos parâmetros passados pelo usuário.	Classe
Application	Interface que define o método que entrega os *bytes* para a aplicação, permitindo assim a transferência de dados.	Interface
Buffers	Usada para melhorar a eficiência do tratamento de mensagens. Provê os métodos para manipulação dos *buffers* de memória.	Classe
BufferTooSmallException	Sinaliza uma exceção lançada ao receber uma mensagem com um *buffer* muito pequeno.	Classe
Connection	Fornece manipuladores de eventos para uma conexão. Também é derivada de Handler.	Classe
Handler	Classe abstrata para administrar conexões.	Classe
Know	Gerencia uma fila de retransmissão que solicita mensagens conhecidas, mas ainda não recebidas.	Classe
Periodic	Classe abstrata que representa uma atividade periódica que pode ser parada e retomada.	Classe
Protocol	Implementação das funções e dados de controle de interface ProtocolMBean.	Classe
ProtocolMBean	Interface de dados de configuração do protocolo, que possibilita o ajuste de parâmetros, a fim de ajustar seu comportamento.	Interface
Queue	Implementação de uma fila com descarte aleatório.	Classe
Queued	Classe abstrata empacotadora para distribuição enfileirada (serial) das informações de mensagem e porta.	Classe abstrata
RandomSamples	Retira amostras aleatórias de um conjunto maior.	Classe

UUIDs	Classe abstrata que manipula (lê e escreve) uma identificação única em um buffer.	Classe abstrata
ConnectionListener	Monitora eventos das conexões.	Interface
DataListener	Monitora os dados trafegados no canal.	Interface
Gossip	Gerencia as técnicas de "fofoca" na disseminação das mensagens.	Classe
MulticastChannel	Lê e escreve dados no canal.	Classe
Overlay	Gerencia os vizinhos de um nó.	Classe
Transport	Responsável pelo envio e recebimento dos pacotes na rede.	Classe

Fonte: Produção do próprio autor.

O estudo do funcionamento das classes descritas na Tabela 2, bem como suas respectivas integrações, foi de fundamental importância para esta pesquisa, ao permitir que o protocolo epidêmico NEEM fosse uma estratégia de comunicação possível de ser utilizada na aplicação CrewFinder, como melhor descrito em 3.2.1.

2.5 Considerações do capítulo

Este capítulo teve como objetivo apresentar o embasamento teórico dos conceitos envolvidos no presente trabalho, desde certos tipos e características de redes sem fio até o surgimento do primeiro conceito de protocolo epidêmico, bem como os parâmetros e funcionamentos desta abordagem e a constatação do dilema, ainda em aberto, quanto ao uso e definição do termo "protocolo", quando envolvido na comunicação epidêmica.

Ainda, foram apresentados com maiores detalhes o *Network friendly epidemic multicast*, sua arquitetura e proposta em ser amigável a rede através da combinação do uso do TCP para camada de transporte e uma técnica de gerenciamento de *buffer*, bem como a justificativa de sua escolha para ser utilizado neste trabalho. Por fim, também foram feitas considerações a respeito da implementação do protocolo na linguagem Java, denominada NeEM *library,* indicando a possibilidade de sua utilização nos experimentos deste estudo.

3. Metodologia

Este trabalho tem como seu principal procedimento, para responder a pergunta-problema que o motivou, o desenvolvimento de uma aplicação para estudos de protocolos epidêmicos, em especial o NEEM. Para tanto, mostrou-se necessário definir as ferramentas, procedimentos, métricas e como foram realizadas cada etapa, bem como a análise dos resultados. Tal discussão encontra-se nas seções seguintes.

Em linhas gerais, para obtenção dos resultados deste trabalho, foram executados 10 experimentos em cada tipo de comunicação proposta (*broadcast* com o protocolo UDP e *multicast-gossip* com o protocolo NEEM). Para tanto, foram utilizados 30 equipamentos reais conectados por uma rede *wireless ad hoc*, trocando mensagens a partir da aplicação construída, na qual apenas a mobilidade das estações era simulada. Para avaliar o desempenho da comunicação, arquivos de *log* eram gerados pela própria aplicação registrando as mensagens recebidas.

3.1 Recursos

Nesta seção são descritos os recursos necessários para a pesquisa como um todo.

3.1.1 Revisão da literatura

A revisão de literatura foi feita ao longo de todo o projeto, utilizando principalmente artigos de bases de dados eletrônicas como ACM (*Association for Computing Machinery*) Digital Library[12], IEEE (*Institute of Electrical and Electronics Engineers*) Xplore Digital Library[13], ScienceDirect (Elsevier)[14], Google Acadêmico[15]. As buscas foram feitas pelas palavras chaves como *manets, gossip, epidemic protocols*.

[12] dl.acm.org
[13] ieeexplore.ieee.org/xplore/home.jsp
[14] www.sciencedirect.com
[15] scholar.google.com.br

3.1.2 Ferramentas

As ferramentas foram necessárias para o desenvolvimento da aplicação, coleta e análise dos resultados. Para a codificação da CrewFinder, a escolha foi pela linguagem Java. Tal escolha justifica-se por:

a) Padronização de ferramentas, por ser a mesma linguagem utilizada pela NeEM *library*;
b) Pensando em uma futura versão da CrewFinder para dispositivos móveis, sendo possível reaproveitar grande parte do código para o sistema operacional Android;
c) Possibilidade de replicação de experimentos por terceiros.

O número de estações escolhidas para realizar o experimento foi de 30. Para chegar a esse número, buscou-se na literatura sobre padrões ou normas que regem sobre o número de socorristas que atendem situações de emergência, porém tal pesquisa foi inconclusiva, visto que varia muito de acordo com o tipo, dimensão e localização da tragédia, com equipes podendo ser compostas por poucos socorristas até grandes esforços internacionais com centenas ou milhares de pessoas. Dessa forma, trazendo o contexto mais próximo da realidade de onde o presente estudo fora desenvolvido, foram levados em consideração dados oficiais do Corpo de Bombeiros do Estado de São Paulo em sua versão mais recente divulgada, do ano de 2017 [34], que pode ser consultado no Anexo A. Dividindo o número total de bombeiros presentes no estado (9.045) pela quantidade de instalações distribuídas pelos municípios paulistas (256) tem-se uma média aproximada de 35 bombeiros por instalação. Além disso, também existiu uma limitação de quantidade dos equipamentos físicos disponíveis para o experimento.

As configurações das 30 estações utilizadas podem ser vistas na Tabela 3:

Tabela 3- Resumo das ferramentas para o experimento

Aplicação	CrewFinder
Plataforma da aplicação	Java
Rede	*Wireless* em modo *ad hoc*
Sistema Operacional das estações	Microsoft Windows 10
Adaptador de rede sem-fio	HP WN7600R-MV 150/300 Mbs 2.4 GHz
Processador	AMD A10-5800B 3800 MHz (modelos 6305) e AMD Phenom II X3 B77 3200 MHz (modelos 6005)
Disco rígido	500 GB
Memória RAM	4 GB
Estações	16 computadores HP Compaq Pro 6305 e 14 computadores HP Compaq Pro 6005

Fonte: Produção do próprio autor.

Cada estação possuía também, além da interface de rede sem fio pela qual ocorria a comunicação *ad hoc* avaliada no experimento, uma interface de rede cabeada com conexão ativa com a Internet. Tal configuração foi necessária para que todas estações mantivessem seus horários sincronizados, essencial para obter as métricas do experimento com precisão. O Windows possui, por padrão, uma implementação simplificada do NTP (*Network Time Protocol*) que não mantém a hora legal brasileira atualizada constantemente. Por isso, foi necessário um programa especializado para este fim, conhecido como *Meinberg NTP Software*[16]. Com isso, todas as estações foram configuradas para obter o horário nos servidores recomendados pelo NTP.BR[17]: a.st1.ntp.br, b.st1.ntp.br, c.st1.ntp.br, d.st1.ntp.br, gps.ntp.br, a.ntp.br, b.ntp.br e c.ntp.br. Tal ação garantiu que os 30 computadores mantivessem a hora, minutos e segundos sincronizados.

[16] Disponível em https://www.meinbergglobal.com/english/sw/ntp.htm

[17] A lista pode ser consultada em https://ntp.br/guia-win-avancado.php#Sincroniza_o_do_Windows_precisa

Para o estudo e análise do tráfego gerado pela aplicação foram coletados *logs*[18] (apêndice A), gerados pela própria CrewFinder em cada estação. Como em cada arquivo desse existem milhares de linhas de informações, mostrou-se necessário uma ferramenta de apoio que fosse capaz de:

a) Importar as informações contidas em cada arquivo de *log* para uma estrutura única;
b) Selecionar, classificar, filtrar e agrupar as informações de diferentes maneiras;
c) Exportar dados facilmente a outros programas para comparação e geração de gráficos.

Para tanto, foi criada uma estrutura para armazenamento e análise dos dados, exibida na Tabela 4:

Tabela 4- Estrutura de armazenamento e análise dos dados

Comparação dos dados e geração de gráficos	Microsoft Excel 2016
Sistema de apoio	MySQL Workbench
Banco de dados	MySQL Community 8.0.15
Sistema operacional	Linux Centos 7
Processador	16 núcleos Intel Xeon
Disco rígido	150 GB
Memória RAM	100 GB
Estação	Dell R720

Fonte: Produção do próprio autor.

A escolha pelo MySQL se deu, principalmente, pelos seguintes fatores: a versão utilizada, Comunnity 8.0.15[19], é um banco de dados *open-source* bastante popular, que conta com grande material de apoio e consulta disponível na Internet. Além disso, dispõe de um completo sistema de apoio ao SGBD (sistema gerenciador de banco de dados), denominado MySQL

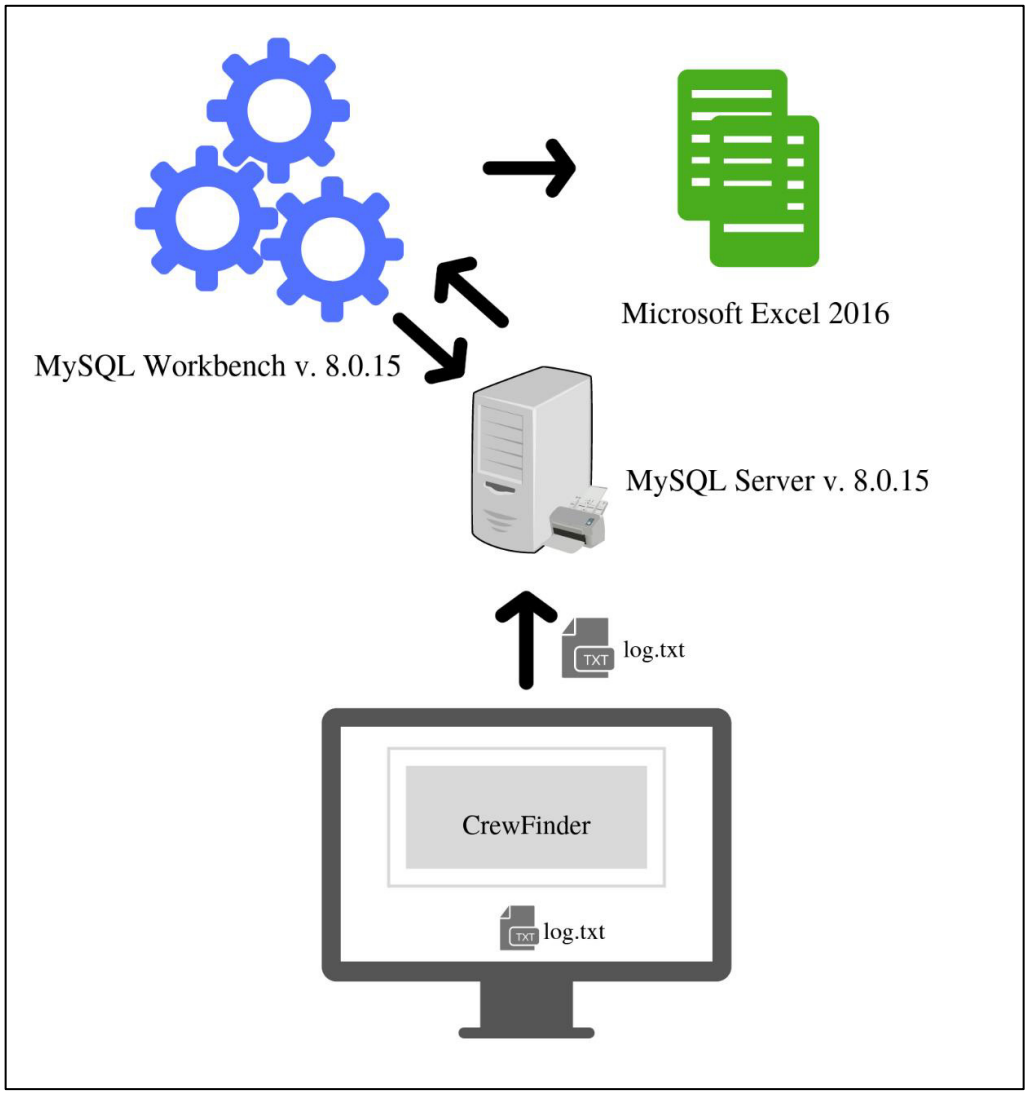
[18] Os *logs* ficarão disponíveis no repositório institucional da Universidade.

[19] Disponível em https://dev.mysql.com/downloads/

Workbench - também gratuito, que foi utilizado tanto na criação da estrutura do banco de dados como na elaboração das consultas para obter as métricas desejadas (disponíveis no apêndice B).

O resultado das consultas realizadas no banco de dados era exportado para uma planilha eletrônica de cálculos, Microsoft Excel 2016, onde foram gerados os gráficos presentes na seção 4 (Resultados e Discussão). Uma ilustração do ambiente utilizado para coleta, armazenamento, consulta e tabulação dos dados pode ser visto na Figura 12.

Figura 12- Representação do ambiente de análise dos resultados



Fonte: Produção do próprio autor.

3.2 Procedimentos

Nesta seção são descritos os procedimentos adotados para o preparo, execução e coleta dos resultados dos experimentos.

3.2.1 Aplicação CrewFinder

Esta etapa constituiu na prototipação, implementação e testes da CrewFinder na linguagem Java. O objetivo da aplicação é informar permanentemente aos membros da equipe a localização vetorizada dos demais integrantes do grupo, de sorte que, na ocorrência de um problema grave com algum deles, seu paradeiro seja conhecido pela equipe. A codificação foi feita no ambiente de desenvolvimento Eclipse[20] e iniciou-se em maio de 2017, após serem obtidos os códigos que foram utilizados em [6] e notar-se que pontos importantes do protocolo NEEM, como o tratamento de *buffer*, não haviam sido implementados de fato.

A estrutura da aplicação foi projetada para organizar cada funcionalidade em sua respectiva classe e abstrair a comunicação com a rede, de modo que diferentes técnicas (*broadcast*, *multicast-gossip,* dentre outras) possam ser implementadas. Assim, o próprio usuário pode escolher com a qual deseja transmitir as informações na rede. Duas implementações desta interface foram feitas para suporte a experimentos: a) uma classe que faz a comunicação através de datagramas usando o protocolo UDP nos modos *unicast* e *broadcast*; b) Comunicação *multicast-gossip* providos pelo protocolo epidêmico NEEM, através de classes provenientes da NeEM *library*. Devido a flexibilidade na estrutura, diferentes técnicas de comunicação podem ser implementadas futuramente, possibilitando que outros estudos possam ser feitos a partir desta aplicação.

Certos parâmetros de comunicação devem ser levados em consideração, já que interferem diretamente nos resultados. Tais parâmetros estão descritos na Tabela 5.

[20] Disponível em https://www.eclipse.org/downloads/

Tabela 5- Parâmetros da aplicação

Payload	Aproximadamente 1 KB como limite máximo.
***Fanout* (para comunicação *multicast-gossip*)**	Quantidade de vizinhos selecionados para receber uma mensagem. Valor 7, com base em [35].
***Hops*/TTL (para comunicação *multicast-gossip*)**	Número de estações que uma mensagem passa até deixar de ser disseminada. Valor 6, padrão da NeEM *library*.
Intervalo de envio de dados	Intervalo, em segundos, que um nó envia uma mensagem com sua localização para a rede. Valor: 2.
Velocidade	Indica a velocidade com que os nós se movimentam no cenário. Foi adotada uma velocidade de até 1.7 metro por segundo, cerca de 6 Km/h.

Fonte: Produção do próprio autor.

Em regiões afetadas por desastres de grandes proporções, existem dificuldades tanto no acesso às áreas isoladas quanto na identificação de sobreviventes e de eventos [36]. Com isso, a mobilidade das equipes de salvamento e resgate nestes ambientes é limitada. Em trabalhos relacionados que envolveram experimentos com MANETs nestas condições, como em [36] e [37], a mobilidade varia entre 1 m/s e 2m/s, chegando no máximo em alguns experimentos a 5 m/s. Portanto, para este trabalho, adotou-se a velocidade máxima de movimentação de até 1,7 m/s, cerca de 6 km/h, um valor adequado pensando na aplicação real do ambiente proposto e condizente com a metodologia adotada nos estudos anteriores.

O tempo de intervalo para envio de dado na rede pela aplicação foi pensado com base na perspectiva do usuário. Como o objetivo desta é manter um mapa de localização com as posições dos membros da equipe, tempos muito longos não atenderiam ao propósito. Então, foi levada em consideração a velocidade máxima de movimentação por segundo (1,7 metros). Partindo de duas situações diferentes: a primeira, considerando duas pessoas próximas, se uma ficar estática e a outra se movimentar, em dois segundos estarão a aproximadamente 3,4 metros de distância uma da outra, o que representa uma distância bem próxima, inclusive para contato físico caso necessário. A segunda situação, se partirem de um mesmo ponto para direções totalmente opostas, em dois segundos poderiam estar a aproximadamente 7 metros distantes uma da outra, que ainda é uma distância razoável para se comunicar, ter visibilidade e até mesmo utilizar dispositivos de segurança, como cordas ou lanternas. Então, chegou-se no entendimento que 2 segundos atenderiam ao propósito da aplicação, como mostra o resultado da Equação 1.

Assim, cada pacote chega em seu limite máximo de tamanho mostrado na Tabela 5 em aproximadamente 15 segundos, que é o tempo necessário para agregar o número de coordenadas configurado.

Equação 1- Cálculo para encontrar distância máxima percorrida no tempo de dado proposto

$$2\ (segundos) \times 1{,}7\ (metros\ por\ segundo) \cong 3{,}4\ (distância\ percorrida\ em\ metros)$$

Fonte: Produção do próprio autor.

Para simular a mobilidade dos nós, primeiramente é calculada a distância máxima que a estação pode ter percorrido. Esse valor é encontrado multiplicando-se a velocidade de movimentação pelo tempo de atualização e por um número real entre 0,0 e 1,0, para simular variações de velocidade. Então, para que ocorra a movimentação, acrescenta-se o resultado obtido nas coordenadas atual em que a estação se encontra. Com isso, a distância máxima que cada nó poderia se mover, considerando a velocidade e tempo, é respeitada. A classe responsável por esses procedimentos é a GPSEmulator e pode ser vista na Figura 13.

Figura 13- Classe GPSEmulator

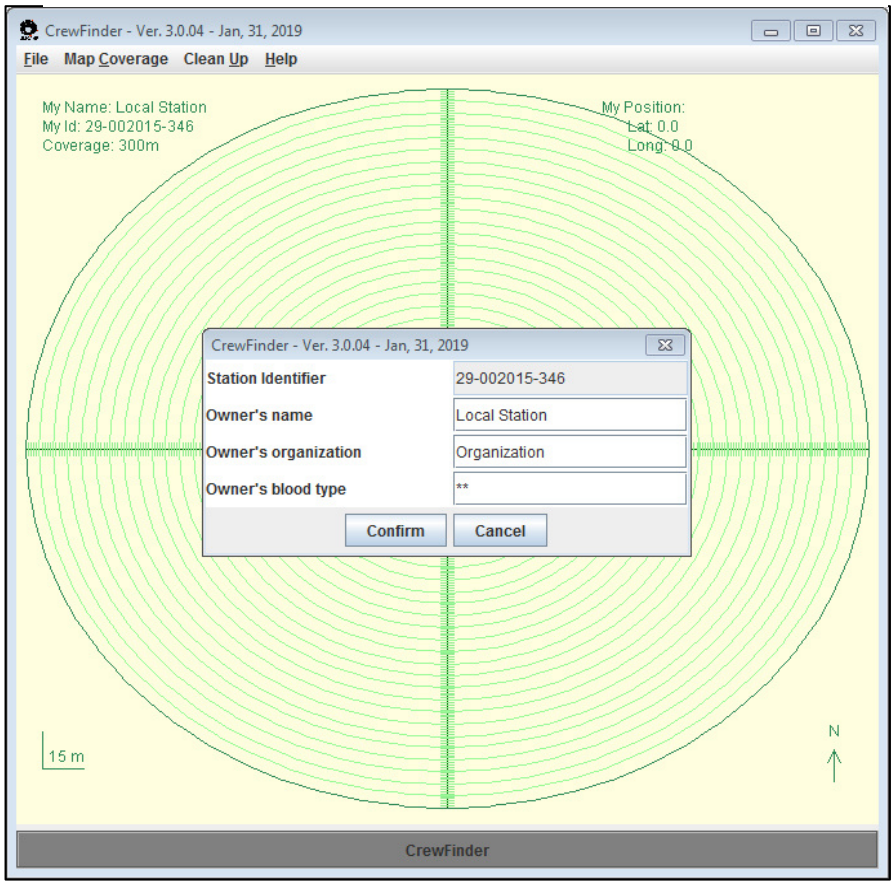
```
public class GPSEmulator
   {
   static private Coordinates currPosition  = new Coordinates();
   static private int         repeatCounter = 0;
   static private boolean     isRepeating   = false;

   public Coordinates readGPS()
      {
      // reads the current position
      Coordinates GPSreading = currPosition;

      if (!isRepeating)
         {
         // prepares the next reading
         double maxDistanceToWalk = (SysPars.getMaxspeed() * SysPars.getTimeToUpdate() / 1000);
         double newPosX = currPosition.getPosX() + ((SysPars.getRandGenerator().nextDouble() * 2 * maxDistanceToWalk) - maxDistanceToWalk);
         double newPosY = currPosition.getPosY() + ((SysPars.getRandGenerator().nextDouble() * 2 * maxDistanceToWalk) - maxDistanceToWalk);
         currPosition = new Coordinates(newPosX, newPosY, System.currentTimeMillis());

         if (SysPars.getRandGenerator().nextInt(20) == 0)
            {
            isRepeating = true;
            repeatCounter = SysPars.getNumCoordToStallCheck() + SysPars.getRandGenerator().nextInt(3);
            }
         }
      else
         {
         currPosition = new Coordinates(currPosition.getPosX(), currPosition.getPosY(), System.currentTimeMillis());
         repeatCounter--;
         if (repeatCounter == 0)
            {
            isRepeating = false;
            }
         }
```

Fonte: Produção do próprio autor.

A divisão da distância máxima por 1000 é feita para obter os valores em segundos. As novas posições das coordenadas X e Y, partindo da posição atual que a estação se encontra, soma-se a um valor aleatório gerado pelo método *randGenerator.nextDouble()*. A multiplicação do valor aleatório por dois e posterior subtração pela distância máxima percorrida, possibilita a existência de valores negativos, ou seja, movimentar-se para coordenadas menores dos que as atuais. Além disso, algumas vezes a mesma posição é repetidamente enviada certo número de vezes para indicar parada de movimento, através da variável "IsRepeating". Para melhor representar uma situação real, foi implementado um filtro para descarte das mensagens recebidas de emissores que estiverem fora da área de alcance do *wireless* (250 metros).

Para iniciar a aplicação, basta executar a classe Start, presente dentro do pacote crewFinderApplication. Então, é exibida a caixa de diálogo mostrada na Figura 14, para preenchimento das configurações do usuário da estação como nome, organização a que pertence e tipo sanguíneo.

Figura 14- Configuração da estação



Fonte: Produção do próprio autor.

Uma vez configurada a estação, ela automaticamente envia sua posição às demais pelo protocolo escolhido no momento da sua partida. A rede é constantemente monitorada e as atualizações das demais estações são recebidas em fluxo contínuo. Cada atualização corresponde a uma "Track", uma classe de dados com a identificação da estação, do usuário, uma estampa de tempo e a lista das últimas coordenadas pelas quais a estação emissora passou. Com base nas atualizações recebidas, a estação compõe o mapa que apresenta ao usuário, tendo a sua própria trilha sempre ao centro e as demais em suas posições relativas. Os dados recebidos passam por um processamento para descarte de duplicidades e mensagens obsoletas, para reclassificação de tipo, no caso, por exemplo, de uma estação que está parada, etc. Um código de cores é usado para indicar ao usuário os tipos das "Tracks" exibidas.

Como a aplicação foi implementada de modo a colocar uma estampa de tempo na geração do pacote que será enviado a rede (independente do protocolo escolhido), consegue-se um controle eficaz, por exemplo, das mensagens recebidas em cada estação, através de análises dos *logs* gerados apenas a nível de aplicação, dispensando verificações diretamente no protocolo de rede em si.

As classes da aplicação foram separadas em pacotes, organizadas de acordo com suas finalidades. São eles: “crewFinderApplication”, “crewFinderDataStructures”, “protocolsPackage”, “crewFinderResources” (contém apenas recursos externos como arquivos de imagens para interface gráfica) e “neemPackage” (já abordado em 2.4).

O Pacote “crewFinderApplication”, descrito na Tabela 6, contém as classes relativas a própria aplicação como interfaces gráficas (janelas, painéis, paleta de cores etc.) e parametrização da aplicação (versão, nome, autores, etc.).

Tabela 6- Classes do pacote CrewFinderApplication

Classe	Descrição	Tipo
BackPannel	Desenha o mapa e as posições das estações.	Classe
ColorPalette	Padronizar a paleta de cores do aplicativo.	Enumeração
ConfigStationDialog	Janela de diálogo para configurar os dados do usuário da estação.	Classe

Controller	Controla as ações do programa, dispara e mantém os *threads* e controla a interface gráfica.	Classe
LogoPanel	Área de desenho para logotipos para composição da interface gráfica.	Classe
MainWindows	Janela principal do sistema.	Classe
MsgScreen	Janela de diálogo para mensagens de uso geral da interface gráfica.	Classe
Start	Partida do aplicativo.	Classe
Sysinfo	Guarda os parâmetros da aplicação para composição da interface gráfica, mensagens e modo de *debug*.	Classe
UtilitaryLibrary	Agrupa métodos de tratamento de *String* e carga de imagem.	Classe
WindowsCustomizedHandler	Classe para resposta ao evento de fechamento da janela principal.	Classe

Fonte: Produção do próprio autor.

Já o pacote crewFinderDataStructures descrito na Tabela 7 agrupa as classes relativas ao problema de localização em si e ao funcionamento do aplicativo em termos de emulação/atualização de posições, *log* de eventos, etc.

Tabela 7- Classes do pacote crewFinderDataStructures

Classe	Descrição	Tipo
ActionType	Categorias de registros do log de atualização.	Enumeração
CFLogger	Classe responsável por gerar os logs de atualização.	Classe
Coordinates	Identifica um ponto unicamente associado ao momento em que foi registrado.	Classe
GPSEmulator	Emula a movimentação do usuário e realiza a leitura do GPS.	Classe
Person	Identifica uma pessoa.	Classe
Station	Identifica unicamente uma estação.	Classe
StationIdentifier	Gera o identificador único para uma estação na rede.	Classe
SysPar	Guarda os parâmetros dos problemas de localização, tais como velocidade máxima, alcance do *wifi,* etc.	Classe
Track	Identifica unicamente um trajeto do usuário.	Classe
TrackType	Define os tipos possíveis de trilhas.	Enumeração

UpdateMyPosition	Implementa um *thread* de envio de atuações de posições para a rede.	Classe
UpdateOtherPositions	Implementa um *thread* que processa as atualizações recebidas da demais estações.	Classe

Fonte: Produção do próprio autor.

No Pacote protocolsPackage, ficam as classes dos protocolos de rede e suas respectivas configurações. A descrição é feita na Tabela 8.

Tabela 8- Classes do pacote ProtocolsPackage

Classe	Descrição	Tipo
ConfigProtocolDialogForNeem	Janela de diálogo para configuração dos parâmetros do protocolo NEEM.	Classe
ConfigProtocolDialogForUDP	Janela de diálogo para configuração dos parâmetros do protocolo UDP.	Classe
NetworkProtocol_Interface	Definição de uma interface para envio e recebimento das atualizações para a rede.	Interface
NetworkProtocol_Neem	Implementação do envio e recebimento de mensagens pelo protocolo NEEM.	Classe
NetworkProtocol_UDP	Implementação do envio e recebimento de mensagens pelo protocolo UDP	Classe
NetworkProtocolFactory	Geração de uma instância específica de uma classe de envio e recebimento de mensagens de um protocolo.	Classe

Fonte: Produção do próprio autor.

Quando a aplicação está em execução, são registrados em um arquivo de texto, dentro do diretório da mesma, informações a respeito das mensagens recebidas pela rede, conforme exemplo que pode ser visto no apêndice A. Cada um dos campos é descrito com detalhes na Tabela 9.

Tabela 9- Estrutura dos arquivos de *log*

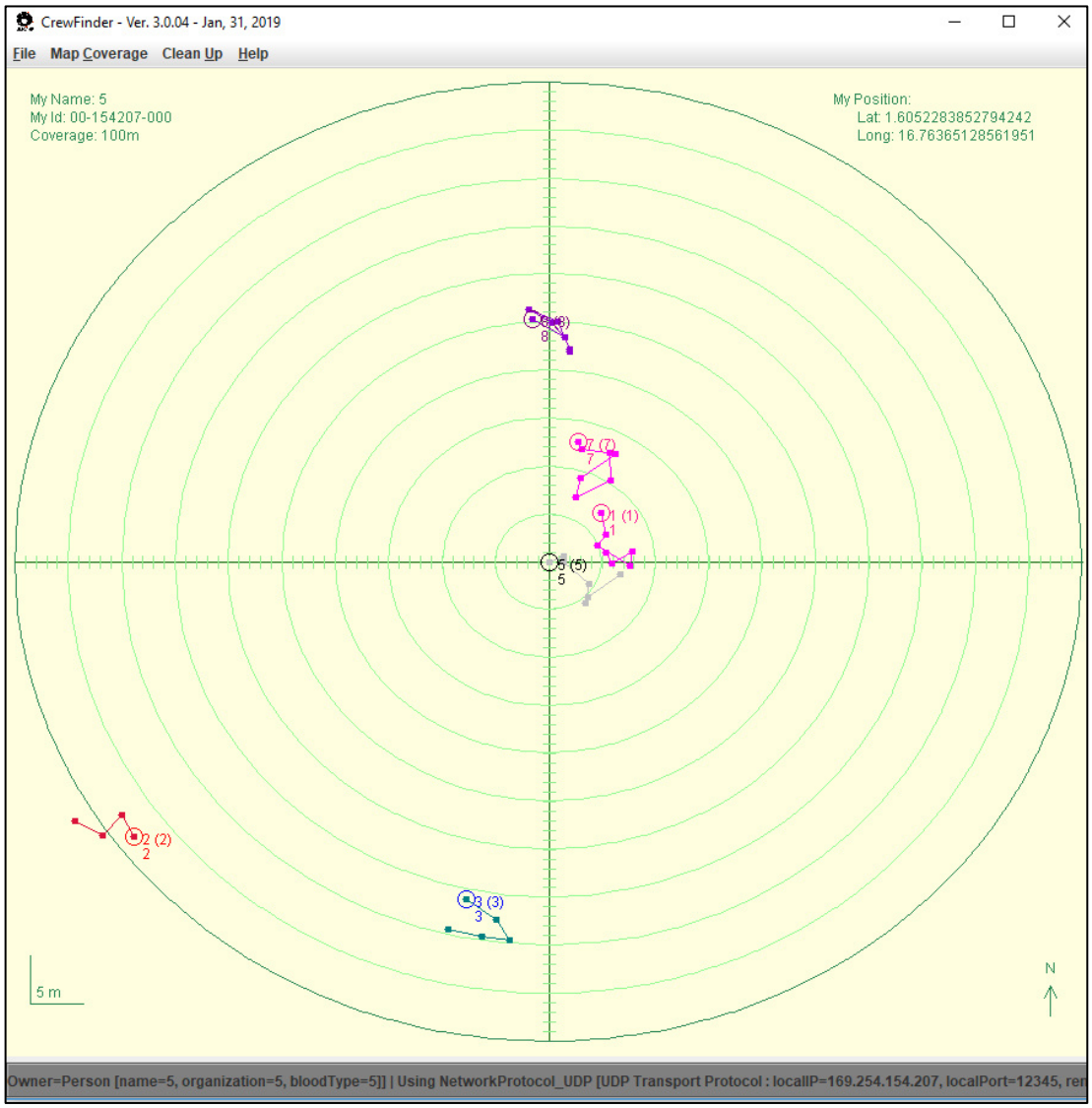
Campo	Descrição	Estado(s) possível(eis)
Identificador	Identificador único da estação que recebeu a mensagem.	
Data	Data e hora que a mensagem foi recebida.	

Processamento	O processamento que a mensagem recebeu na estação.	"**Just Received**": registra apenas que uma mensagem foi recebida, "**Reclassified**": atualização da posição de alguma estação conhecida, "**Own track**": trilha da própria estação, "**Disappeared**": estações conhecidas que deixaram de enviar suas posições, "**Old or dupplicated**": trilha desatualizada de alguma estação conhecida ou mensagem já recebida anteriormente, "**Out of range**": mensagens fora de alcance são descartadas como se não tivessem sido recebidas, mas são registradas no log.
Ação	Ação tomada a com mensagem pela estação após recebe-la.	"**RECEIVED**": registra apenas que uma mensagem foi recebida, "**ACCEPTED**": a mensagem foi válida para a estação, ou seja, não era antiga, duplicada nem dela mesma, "**CHANGED**": mudança de um status prévio, "**DISCARDED**": mensagem antiga, duplicada, recebida dela mesma ou de alguma estação que esteja fora do alcance da rede, portanto foi descartada.
ID da estação emissora	Identificador da estação que foi a emissora primária.	
Status	De qual tipo se refere a trilha da mensagem recebida. Cada tipo foi programado para ser exibido com diferentes cores na tela da aplicação, para facilitar a compreensão sobre o que está acontecendo naquele exato momento.	"**MYTRACK**": trilhas da própria estação, "**NORMAL TRACK**": trilhas em "situação normal" de outras estações, "**STALLEDTRACK**": trilha de estações que não se movimentaram (ou que não enviam mensagens há pouco tempo, que posteriormente passam a constar como desaparecidas) e "**EXITINGTRACK**": trilha de estações que estão notificando sua saída da aplicação.
Criação	Momento em que a mensagem foi gerada.	
Trilhas	Parâmetro que define a quantidade máxima de trilhas contidas em uma mensagem (valor 7 no caso deste trabalho, porém pode ser alterado).	
Trilha 1 (até 7)	Para cada trilha: número serial, momento em que foi gerada e coordenadas X e Y.	

Fonte: Produção do próprio autor.

Na Figura 15 é possível visualizar um exemplo de cada descrição do campo “Status”. A trilha cinza (estação 5) é do tipo “MYTRACK”, a azul (estação 3) é do tipo “NORMALTRACK”, a vermelha (estação 2) é do tipo “STALLEDTRACK”, a roxa (estação 8) é do tipo “EXITINGTRACK” e as rosas (estações 1 e 7) são do tipo “DISAPPEAREDTRACK”. Uma estação é considerada desaparecida por outra se sua posição deixar de ser recebida por 3 atualizações consecutivas (6 segundos).

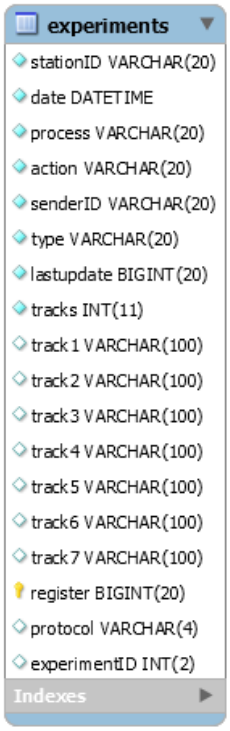
Figura 15- Diferentes tipos de trilhas apresentadas na tela da CrewFinder



Fonte: Produção do próprio autor.

Como mencionado em 3.1.2, foi necessário criar um banco de dados para armazenar os *logs*. Assim, cada parte da estrutura da Tabela 9 virou um campo na tabela no banco de dados, conforme pode ser visto na Figura 16.

Figura 16- Estrutura da tabela no banco de dados



Fonte: Produção do próprio autor.

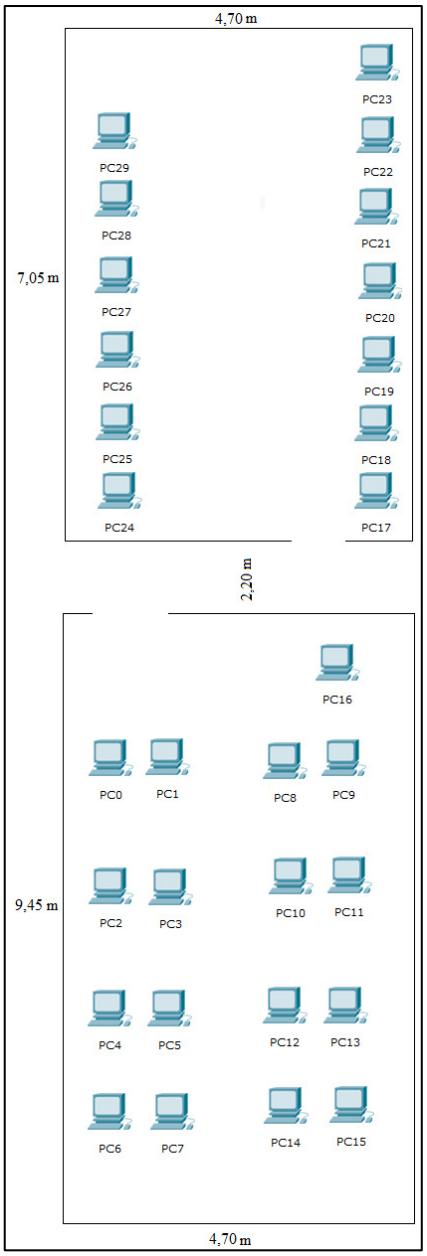
A estrutura dos arquivos de *logs* projetada para a CrewFinder provê o registro de uma série de informações, através das quais é possível chegar nas métricas esperadas para as conclusões deste trabalho e até mesmo em outras não exploradas neste, mas que podem ficar como sugestão de estudos futuros.

3.2.2 Preparação do ambiente e execução dos experimentos

Optou-se por realizar 10 experimentos para cada tipo de comunicação (*multicast-gossip* através do protocolo NEEM e *broadcast* através do protocolo UDP), com duração de uma hora cada. Com o total de 20 experimentos, foi possível obter resultados comparativos que permitiram chegar às conclusões deste trabalho.

Os 30 equipamentos utilizados ficaram divididos em 2 ambientes físicos diferentes. Tratam-se de duas salas com dimensões aproximadas de 33m² e 44m², construídas de alvenaria e separadas por um corredor de 2.20 metros. Uma ilustração representativa do ambiente experimental pode ser vista na Figura 17.

Figura 17- Ilustração do ambiente experimental



Fonte: Produção do próprio autor.

A configuração da rede sem fio em modo *ad hoc*, utilizada nos experimentos, seguiu a padronização de endereços IPv4 172.16.0.0/24, conforme a Tabela 10.

Tabela 10- Endereçamento IP do ambiente experimental

Estação	Endereçamento IP	Estação	Endereçamento IP
PC0	172.16.0.100/24	PC15	172.16.0.115/24
PC1	172.16.0.101/24	PC16	172.16.0.116/24
PC2	172.16.0.102/24	PC17	172.16.0.117/24
PC3	172.16.0.103/24	PC18	172.16.0.118/24
PC4	172.16.0.104/24	PC19	172.16.0.119/24
PC5	172.16.0.105/24	PC20	172.16.0.120/24
PC6	172.16.0.106/24	PC21	172.16.0.121/24
PC7	172.16.0.107/24	PC22	172.16.0.122/24
PC8	172.16.0.108/24	PC23	172.16.0.123/24
PC9	172.16.0.109/24	PC24	172.16.0.124/24
PC10	172.16.0.110/24	PC25	172.16.0.125/24
PC11	172.16.0.111/24	PC26	172.16.0.126/24
PC12	172.16.0.112/24	PC27	172.16.0.127/24
PC13	172.16.0.113/24	PC28	172.16.0.128/24
PC14	172.16.0.114/24	PC29	172.16.0.129/24

Fonte: Produção do próprio autor.

Primeiramente, na estação “PC0”, era criada a rede *ad hoc* no adaptador de rede sem fio, através da execução dos seguintes comandos (os parâmetros após “ssid” e “Key” referem-se ao nome da rede e senha para acesso, respectivamente):

Netsh Wlan Set hostedNetwork Mode=Allow ssid=AD-HOC Key=p@ssw0rd

Netsh Wlan Start hostedNetwork

Posteriormente, as demais estações eram manualmente conectadas na rede criada “AD-HOC”. Após as configurações de conexão acima, um teste de conectividade com pacotes ICMP através do comando *ping* era disparado para todas as estações, para certificar a operacionalidade desta e, na ocasião de algum problema, resolvê-lo antes do início do experimento. Este procedimento garantiu a disponibilidade da rede independente do protocolo a ser testado naquele momento. Feitas as devidas configurações e checagens, era então iniciada a aplicação

CrewFinder em todas as estações, uma por vez, através do IDE (*Integrated Development Environment*) Eclipse.

Nos testes em que se pretendia utilizar a comunicação *broadcast* através do protocolo UDP, para iniciar a CrewFinder, eram informados os parâmetros descritos na Tabela 11 para a classe Start.

NetworkProtocol_UDP 12345 176.16.0.255 12345

Tabela 11- Parâmetros da aplicação para o protocolo UDP

Parâmetro	Significado
NetworkProtocol_UDP	Classe responsável pela comunicação UDP.
12345	Porta de comunicação a ser aberta na estação
172.16.0.255	Endereço de broadcast envio/recebimento dos pacotes
12345	Porta de comunicação do destinatário

Fonte: Produção do próprio autor.

Nos testes em que se pretendia utilizar a comunicação *multicast-gossip* através do protocolo NEEM, para iniciar a CrewFinder, eram informados os parâmetros descritos na Tabela 12 para a classe Start.

NetworkProtocol_Neem 12302 172.16.0.100:12301

Tabela 12- Parâmetros da aplicação para o protocolo NEEM

Parâmetro	Significado
NetworkProtocol_Neem	Classe responsável pela comunicação epidêmica NEEM.
12302	Porta de comunicação a ser aberta na estação
172.16.0.100:12301	Endereço e porta de alguma estação da rede que já esteja se comunicando através do NEEM.

Fonte: Produção do próprio autor.

O tempo começava a ser cronometrado somente quando as 30 estações estivessem com a aplicação em execução, para garantir uma hora de comunicação com todos os computadores. Esgotado este tempo, a aplicação era encerrada. Posteriormente, em cada uma das estações, era verificado a integridade do arquivo e coletado o *log* produzido durante o teste.

Antes e também durante cada experimento, os relógios eram constantemente verificados para garantir a integridade da sincronia (em caso de diferença de segundos, por exemplo, o experimento era descartado), conforme relatado em 3.3.

3.3 Lições aprendidas com os experimentos iniciais

No total, além dos 20 testes pretendidos, foram executados cerca de 10 a 15 ensaios descartados ao longo do processo experimental devido fatores externos indesejáveis como queda de energia, falhas de hardware/software ou até mesmo perda de sincronia entre os relógios das estações. Qualquer evento que comprometesse ao menos uma estação ou a comunicação em si fazia com que o experimento fosse desconsiderado, pois o resultado geral seria comprometido. Com isso, garantiu-se que os dados analisados para obtenção das conclusões deste trabalho fossem provenientes de experimentos íntegros, de acordo com a metodologia proposta.

A título de exemplo, nos primeiros testes com o protocolo epidêmico NEEM foi constatado que seria necessária uma mudança no parâmetro *fanout* testado com o valor 11 (padrão da NeEM *library).* Tal mudança foi necessária pois os relógios das estações perdiam a sincronia em poucos minutos de experimento, fato que não acontecia quando era utilizado o protocolo UDP. Notou-se então que a rede estava muito sobrecarregada e buscou-se na literatura possíveis causas, quando foi encontrado o trabalho [35], que utilizou o protocolo NEEM em uma rede com exatamente 30 nós e obteve resultados satisfatórios com o valor de *fanout* 7. Nesse mesmo estudo foi concluído que os parâmetros da comunicação epidêmica estão diretamente ligados ao desempenho da rede. Isto indica que uma futura análise da sensibilidade dos parâmetros pode ser interessante para este trabalho, assim como variações na quantidade de estações.

Outra alteração que se mostrou válida por experimentos primários foi em relação ao tempo de intervalo para envio de dados pela aplicação. Inicialmente, pensou-se no valor de 30 segundos, baseado no alcance *wireless* de 250m em ambientes externos (considerando a

situação em que uma estação esteja estática e outra em movimento, levaria aproximadamente 2 minutos e 30 segundos para que não estivessem mais dentro da mesma área de alcance, levando em conta a movimentação na ordem de 1,7 m/s e, caso partissem de um mesmo ponto se movimentando em direção opostas, levaria aproximadamente 1 minuto e 15 segundos para que perdessem a comunicação). Dessa forma, considerando uma margem de segurança, tinha sido adotado o tempo de envio de dados de 30 segundos. Porém, pensando que o sentindo da aplicação é manter um mapa constantemente atualizado das posições, o tempo de 30 segundos era bastante longo, inclusive para o desenvolvimento e testes da própria aplicação, já que poucas mensagens eram geradas. Então, pensou-se que faria mais sentido definir este tempo na perspectiva do usuário, como mostrado em 3.2.1.

Ainda, como os laboratórios utilizados nos experimentos deste trabalho são ambientes acadêmicos, a disponibilidade para testes fora aos finais de semana e feriados, tornando esta etapa da pesquisa mais demorada que o esperado já que, além do período de 1 hora do experimento em si, era necessário tempo para os procedimentos anteriores (como configuração de rede e aplicação das 30 estações) e posteriores (coleta dos *logs*). Por isso, mostrou-se necessário criar uma definição bem clara de tais procedimentos, pois além de garantirem a integridade dos testes, guiaram o fluxo do trabalho experimental de forma a otimizar o tempo.

3.4 Dificuldades encontradas

Uma das principais dificuldades encontradas neste trabalho, que demandou uma quantidade considerável de tempo para ser contornada, foi ao iniciar a revisão bibliográfica e se deparar com inconsistências na literatura sobre a terminologia “protocolo”, amplamente adotada na comunicação epidêmica. Notou-se a inexistência de uma padronização sobre elementos que caracterizam de fato o que seja ou não um protocolo epidêmico e que investigações mais profundas sobre o tema abririam uma nova possibilidade de pesquisa, ao mesmo tempo que tiraria o propósito inicial desta. Portanto, optou-se por manter o foco na análise do desempenho e apenas enfatizar a existência do dilema sobre a terminologia, trazendo considerações do autor do NEEM sobre o tema, bem como sugerir estudos futuros a respeito.

Definido o foco, novas inconsistências na literatura surgiram, em especial na adotada como principal base teórica e metodológica desse trabalho. Discrepâncias entre o texto e a

implementação apontaram que o emprego dos códigos simulados poderia resultar na produção de resultados não confiáveis. Tal constatação novamente desacelerou o andamento do trabalho, já que foi necessária uma total reestruturação na metodologia. Com isso, a principal preocupação passou a ser a produção de resultados confiáveis para um ambiente real, através de experimentos em equipamentos e ambiente de rede físicos, fazendo a medição e análise do tráfego gerado por uma aplicação construída adequadamente.

Com a proposta de fornecer resultados confiáveis para o ambiente de MANETs, mostrou-se necessário definir um cenário que, além de realista, fosse viável para ser empregado nos experimentos. Um certo tempo foi necessário para tal definição, já que se buscou na literatura até contatar-se que este tipo de rede vem sendo estudada em pesquisas como uma interessante alternativa de comunicação para ambientes de desastre.

Definido o cenário, o próximo desafio que demandou uma reflexão cautelosa foi pensar em uma aplicação que se mostrasse com certa utilidade considerando o contexto, além de viável de ser desenvolvida. A ideia da CrewFinder foi originada em fornecer apoio a usuários que se beneficiariam de seu uso no cenário considerado de MANETs. Como o trabalho já estava em andamento, o tempo se mostrou um fator bastante preocupante.

Além do desenvolvimento sobre padrões e protocolos adequados, foi necessário implementar uma interface flexível de comunicação com a rede de forma que o protocolo de comunicação pudesse ser livremente alterado pelo usuário, justamente para permitir a avaliação de desempenho proposta. Ainda, para tornar esse requisito funcional no caso do *multicast* epidêmico NEEM, foi necessária a integração com sua biblioteca já existente, demandando estudos aprofundados a respeito do funcionamento da mesma.

Como já mencionado, a possibilidade de medições e análise do tráfego gerado também era um requisito importante. Para tanto, a aplicação foi programada para registrar dados como identificação da estação e uma estampa de tempo em cada pacote gerado. Dessa forma, foi possível gravar informações a respeito das mensagens recebidas em um arquivo de *log*, afim de obter as métricas pretendidas.

Outro desafio foi, justamente, a disponibilidade de equipamentos físicos para os experimentos. Primeiramente, houve dificuldades em encontrar padrões sobre equipes de resgate na literatura, como número de socorristas por exemplo, já que isso tinha influência direta com o número de computadores necessários para os experimentos. Posteriormente, ao chegar-se no entendimento de que 30 estações seria um número razoável para o contexto proposto, notou-se que Faculdade não dispunha dessa estrutura disponível. Para manter a proposta, a alternativa foi buscar uma parceria com o Instituto Federal de São Paulo Câmpus Capivari, que dispôs de 2 laboratórios aos finais de semanas e feriados para realização dos testes.

Ainda, experimentos em laboratório apresentam um desafio extra. Apesar de tratar-se de um ambiente controlado, estão mais suscetíveis a uma série de fatores externos quando comparados a simuladores. Além disso, todas as configurações a nível de aplicação, sistema operacional e rede precisam ser feitas em cada uma das estações e falhas que ocasionalmente ocorram em uma delas comprometem o experimento como um todo.

Pensando no propósito da aplicação, a qual apresenta característica de mobilidade, um novo obstáculo foi encontrado. Os equipamentos disponíveis para os testes tratavam-se de *desktops*, dos quais recursos de GPS não eram disponíveis. Além dessa limitação em termos de recursos, também não foi encontrado um referencial teórico que pudesse servir como metodologia para embasar uma possível mobilidade real. A solução foi implementar na aplicação uma classe responsável por fazer a simulação da movimentação, através de um algoritmo de caminhada aleatória, já bastante conhecido na literatura.

Dada tamanha reformulação na metodologia, foi necessário um grande esforço para viabilizar a proposta do trabalho, diante das múltiplas tecnologias e ambientes envolvidos (desde linguagens de programação como Java e SQL, conhecimentos técnicos para configuração de servidores e redes, além da disponibilidade/utilização de infraestrutura externa à Faculdade).

Para operacionalizar cada etapa, procedimentos que à primeira vista pareciam triviais acabaram mostrando-se de certa complexidade, apresentando mais problemas do que o esperado. Tais dificuldades muitas vezes acabaram demandando estudos mais aprofundados que por consequência estenderam o prazo do trabalho previsto inicialmente.

3.5 Indicadores

Para avaliar o desempenho do ambiente proposto as seguintes métricas foram analisadas, através dos *logs* coletados:

a) Quantidade de mensagens descartadas: avalia a quantidade de mensagens redundantes, desatualizadas, recebidas pela própria emissora ou fora de alcance;
b) Quantidade de mensagens válidas: avalia a quantidade de mensagens recebidas pelas estações, desconsiderando as descartadas;
c) Latência de atualização: avalia o tempo máximo entre o envio de uma mensagem pelo nó transmissor até o último receptor;
d) Taxa de nós atingidos na rede: avalia o quanto uma mensagem válida foi disseminada por meio da quantidade de estações atingidas;

Tais métricas podem também ser comparadas com as obtidas em [6], visto que foram observados, dentre os muitos cenários presentes naquele estudo, alguns parecidos com o proposto nesta pesquisa (32 nós se movendo a velocidade média de 8 km/h, inclusive com o *fanout* 7), porém com o uso de simulador. Os resultados podem ser vistos na Tabela 13. Apesar de já pontuado que os números daquele estudo podem não representar o comportamento de uma aplicação real, comparações com o presente trabalho, que procurou construir uma aplicação e condições experimentais mais realista dentro do possível, trazem discussões interessantes a respeito de ambos estudos.

Tabela 13- Resultados dos experimentos em cenários de baixa velocidade

Fanout	**Latência (s)**	**% de mensagens perdidas**	**% nós atingidos**	**Média de mensagens duplicadas**
5	0,102380	8,49	91,24	2
6	0,118740	5,25	94,58	3
7	0,119827	2,99	96,91	4
8	0,111513	1,78	98,16	5
9	0,122663	1,11	98,85	5
10	0,122327	0,67	99,31	6

Fonte: Retirada de [6].

Os resultados mostrados são satisfatórios. É possível observar que para todos os cenários as latências são abaixo de um segundo, além da diminuição gradual na taxa de mensagens perdidas aumentando-se o *fanout,* resultando também em maior cobertura de nós atingidos e número de mensagens duplicadas. Tais resultados deixam em evidência o mecanismo de funcionamento da comunicação epidêmica: alcança-se melhores taxas de entrega e menores perdas aumentando a retransmissão que, como consequência, resulta em maior número de mensagens duplicadas.

3.6 Considerações do capítulo

Este capítulo buscou descrever, com detalhes, todos os procedimentos necessários bem como as práticas adotadas para chegar nos resultados. Foram apresentadas todas as ferramentas empregadas, desde bases científicas para consultas, linguagens e ambientes de programação até os indicadores esperados.

Em especial, foi feita uma abordagem sobre a CrewFinder, sua estrutura de classes, funcionamento e características importantes como a interface flexível de comunicação com a rede, integração com a NeEM *library* e geração de *logs* que possibilitaram análises a nível de aplicação. Também foi explicado sobre o único recurso de simulação utilizado na implementação da classe que emula uma caminhada aleatória para mobilidade das estações.

Ainda, de maneira bem definida, foram explanados todos os procedimentos adotados na etapa experimental, de modo a facilitar futuras reproduções por terceiros, inclusive com relatos de lições aprendidas e principais dificuldades encontradas.

4. Resultados e Discussão

Os resultados apresentados nesta seção foram extraídos através de consultas às milhões de linhas de *logs* geradas pelas estações ao longo dos experimentos. Como a proposta era comparar o desempenho da comunicação *multicast-gossip* com a *broadcast* em redes *ad hoc* móveis, optou-se por mostrar tais comparações através de cada métrica definida na seção 3.5, além de outras informações que se mostraram relevantes durante o processo de análise dos resultados.

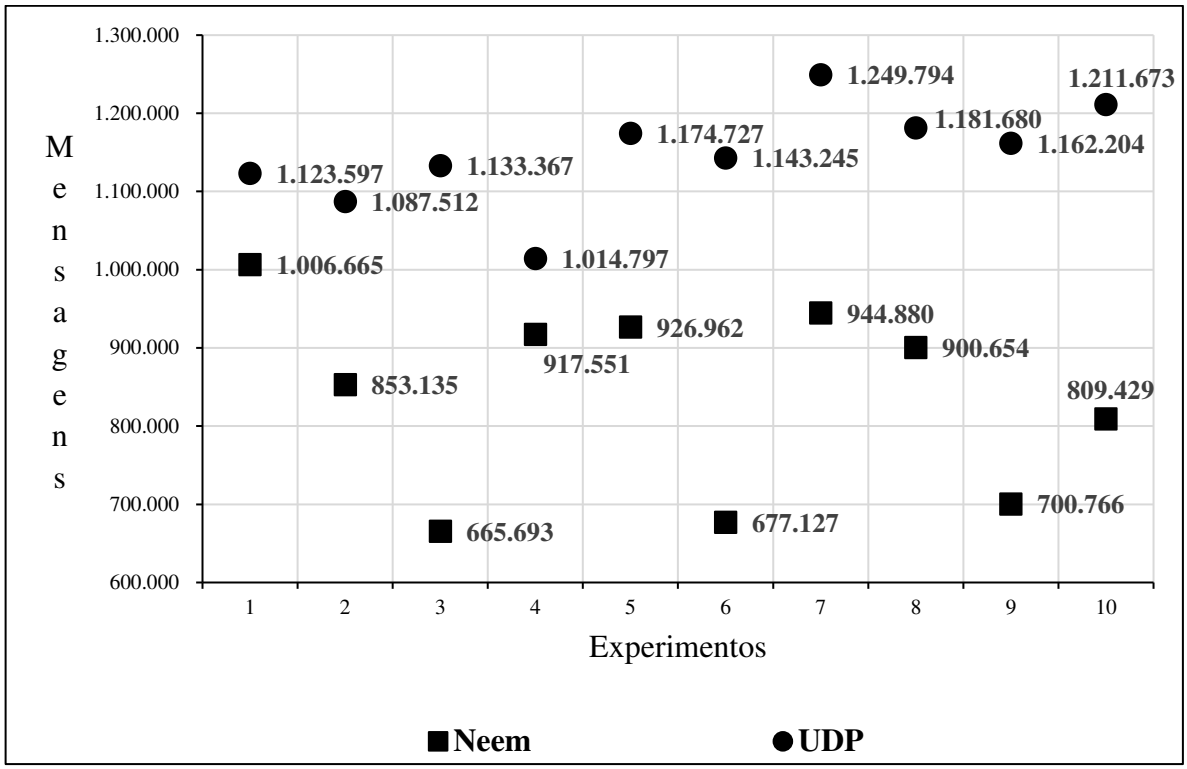
4.1 Métricas obtidas

Para facilitar a compreensão dos resultados, adotaram-se os seguintes termos:

- Mensagens fora do alcance: mensagens que em uma situação real seriam perdidas por falta de alcance *wireless*, mas que nos experimentos puderam ser contabilizadas dadas as condições de simulação da movimentação;
- Mensagens totais: Não contabiliza as mensagens fora de alcance;
- Mensagens descartadas: Mensagens que foram recebidas mais de uma vez, recebidas desatualizadas ou as que foram recebidas pela própria estação emissora;
- Mensagens válidas: São as mensagens totais subtraindo as mensagens descartadas.

A primeira observação fica por conta da quantidade de mensagens totais, que pode ser vista na Figura 18. Nos experimentos executados sobre a comunicação *multicast-gossip*, menos mensagens são recebidas pelas estações quando comparado aos experimentos sobre a comunicação *broadcast* com o protocolo UDP, levando em conta que o padrão de envio de dados da aplicação foi o mesmo em todos os testes.

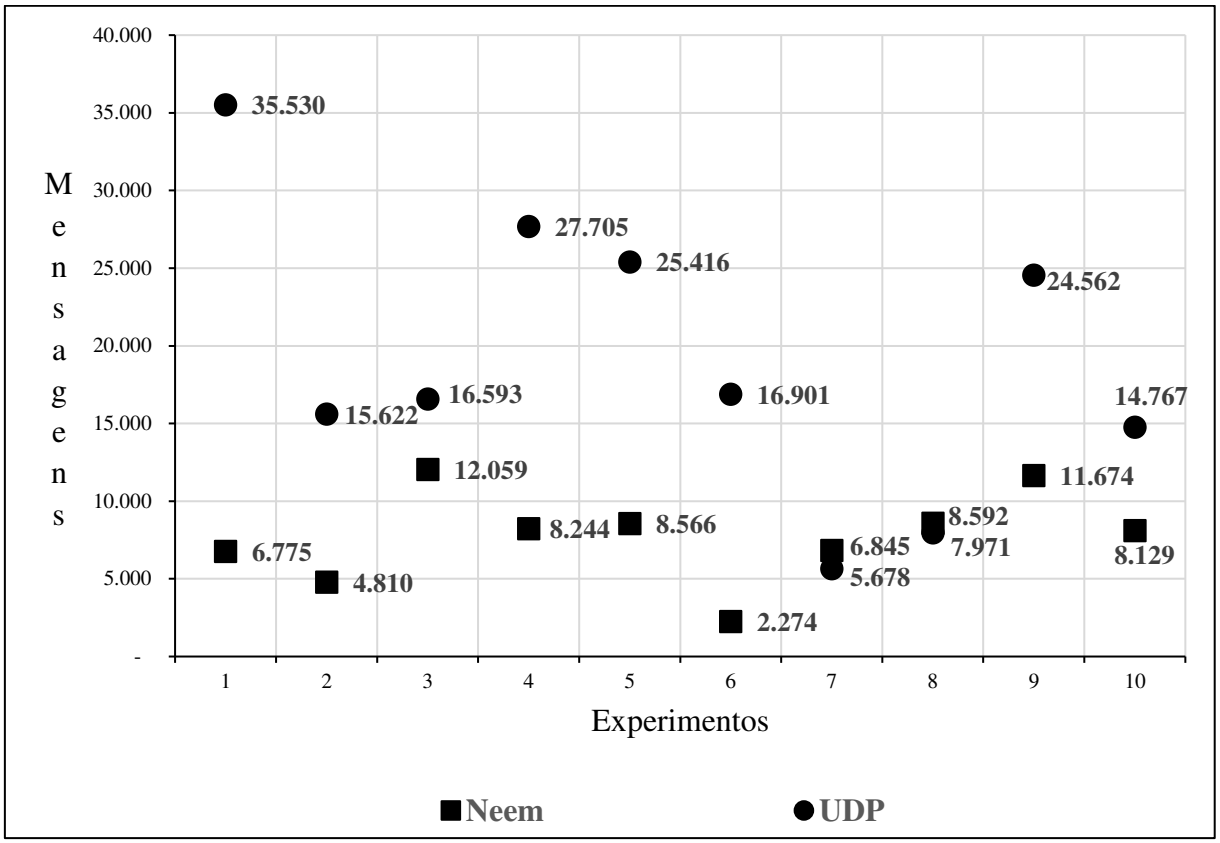
Figura 18- Mensagens totais recebidas em cada experimento



Fonte: Produção do próprio autor.

As estações recebem menos mensagens nos testes com o NEEM. Os parâmetros da comunicação epidêmica como o *fanout,* utilizados para controlar a disseminação, acabam limitando para quantas estações as mensagens são enviadas. Já no *broadcast,* o envio é endereçado para toda rede, sem nenhum limitador. Além disso, não eram esperadas perdas significativas na rede em nível físico, já que se tratava de um ambiente de testes controlado. O total de mensagens fora de alcance pode ser visto na Figura 19.

Figura 19- Total de mensagens fora de alcance em cada experimento

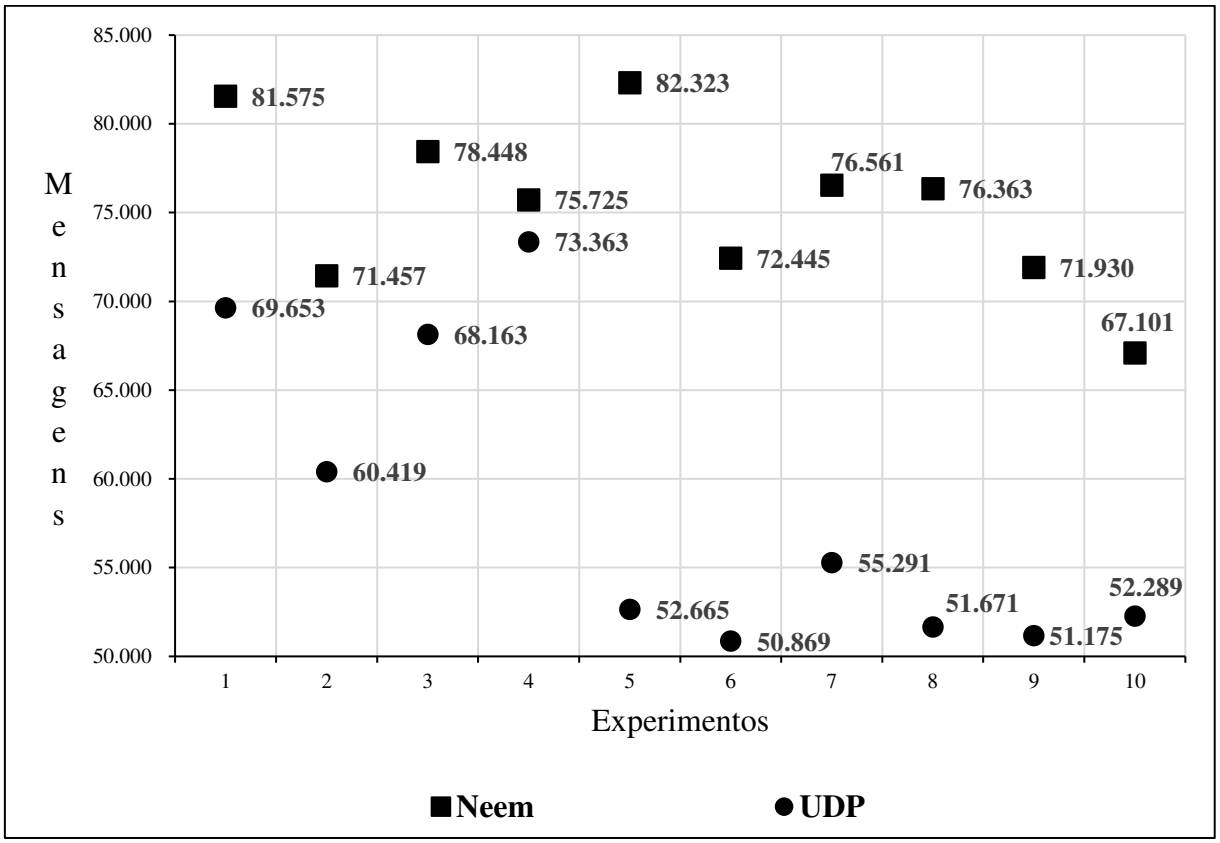


Fonte: Produção do próprio autor.

Nota-se uma diferença entre os protocolos, onde as estações comunicando-se sobre o UDP apresentam maior recebimento de mensagens fora de alcance, com exceção dos experimentos 7 e 8, nos quais por uma diferença pequena a situação é inversa. Vale considerar que o padrão de movimentação e duração foram os mesmos em todos os experimentos para ambos protocolos. Dada igualdade nas condições experimentais, a diferença pode ser atribuída a capacidade de entrega de cada protocolo, como já observado na Figura 18.

O UDP é mais vantajoso nas métricas de mensagens descartadas ao apresentar menores as taxas, como pode ser visto na Figura 20. O protocolo epidêmico NEEM, além de entregar menos mensagens para as estações, apresenta maior taxa de descarte.

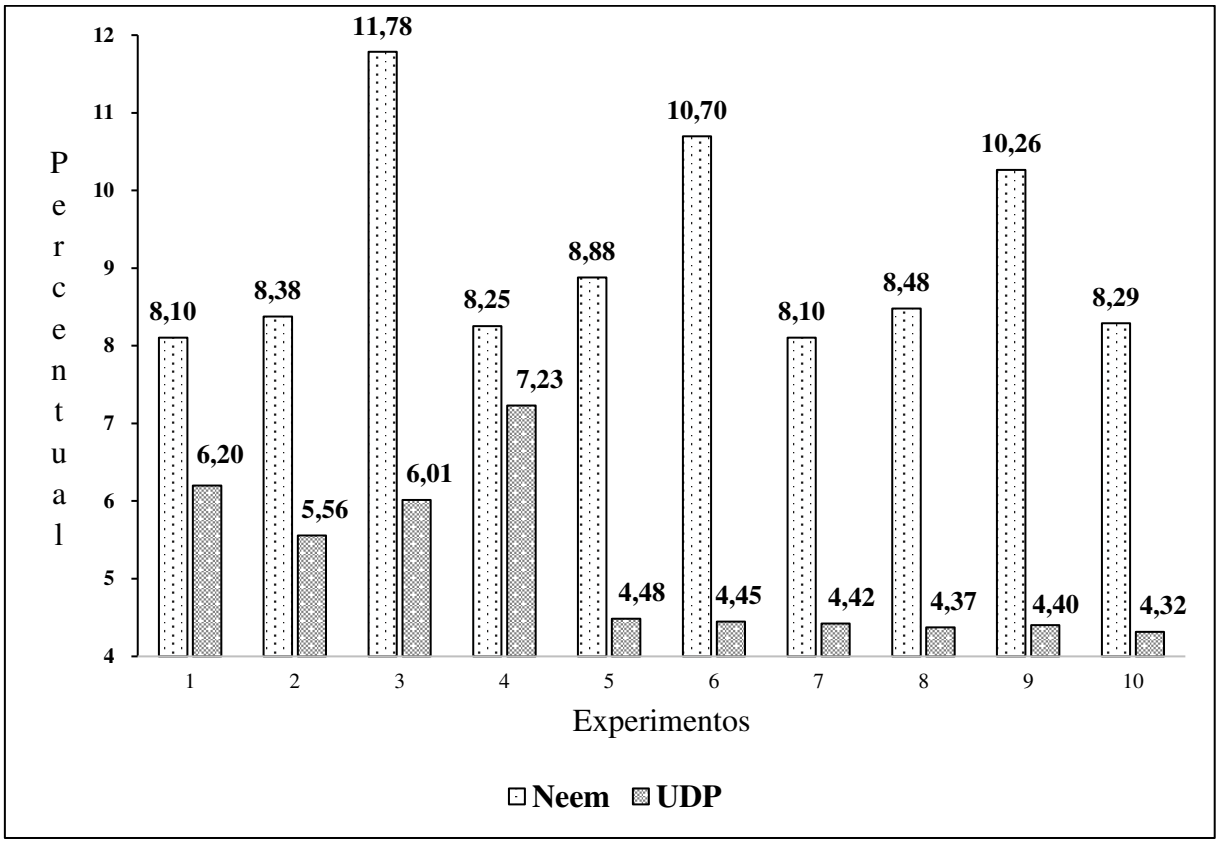
Figura 20- Mensagens descartadas em cada experimento



Fonte: Produção do próprio autor.

Na Figura 21 é exibida a porcentagem das mensagens descartadas em relação aos totais em cada forma de comunicação. O fato das estações se comunicando através do NEEM descartarem praticamente o dobro na maior parte dos casos pode ser explicado pela própria natureza da comunicação *multicast-gossip,* em que as mensagens ficam sendo replicadas na rede.

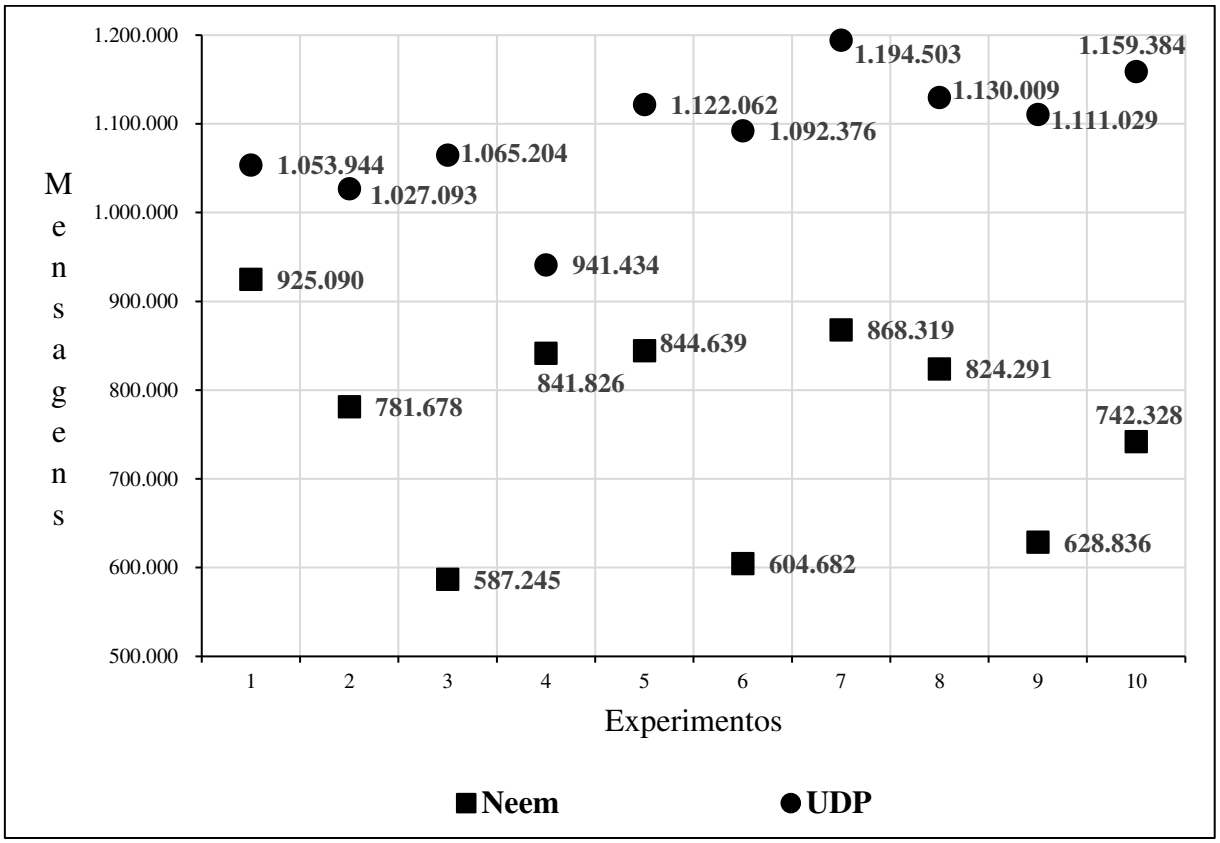
Figura 21- Percentual de mensagens descartadas por experimento



Fonte: Produção do próprio autor.

A Figura 22 mostra que o UDP entrega mais mensagens válidas. Elas são importantes porque carregam as atualizações mais recentes de cada emissor, sendo fundamentais para o objetivo da aplicação. É desejável que atinjam o maior número possível de estações.

Figura 22- Mensagens válidas

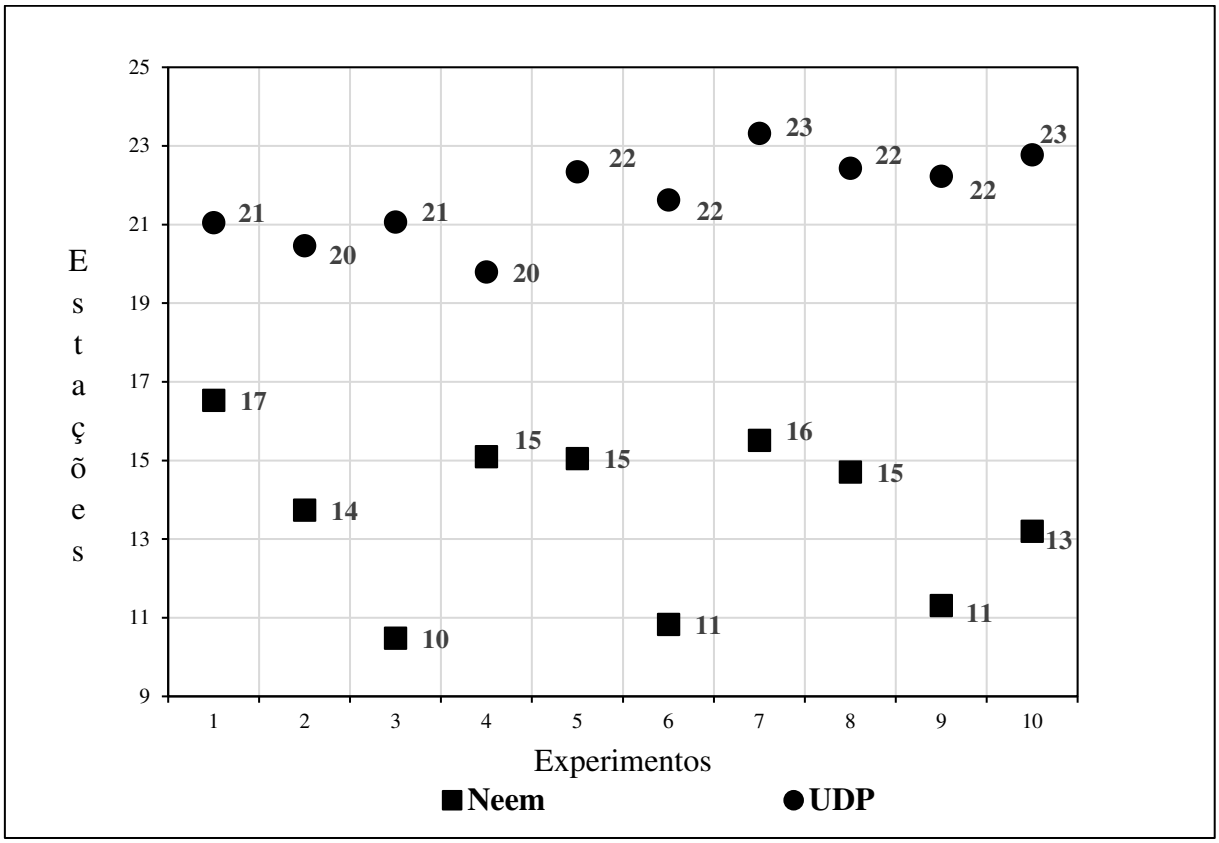


Fonte: Produção do próprio autor.

O resultado observado na Figura 22 mostra que os mecanismos de gerenciamento providos pelo protocolo NEEM na comunicação *multicast-gossip* não se mostraram benéficos para este cenário, pois esperava-se que mensagens válidas fossem priorizadas semelhante à sua validação experimental em [20].

Ainda, a Figura 23 mostra que a comunicação com o protocolo UDP novamente apresenta melhor desempenho na cobertura da rede com mensagens válidas, o que não difere dos resultados já apresentados.

Figura 23- Média de estações atingidas com mensagens válidas

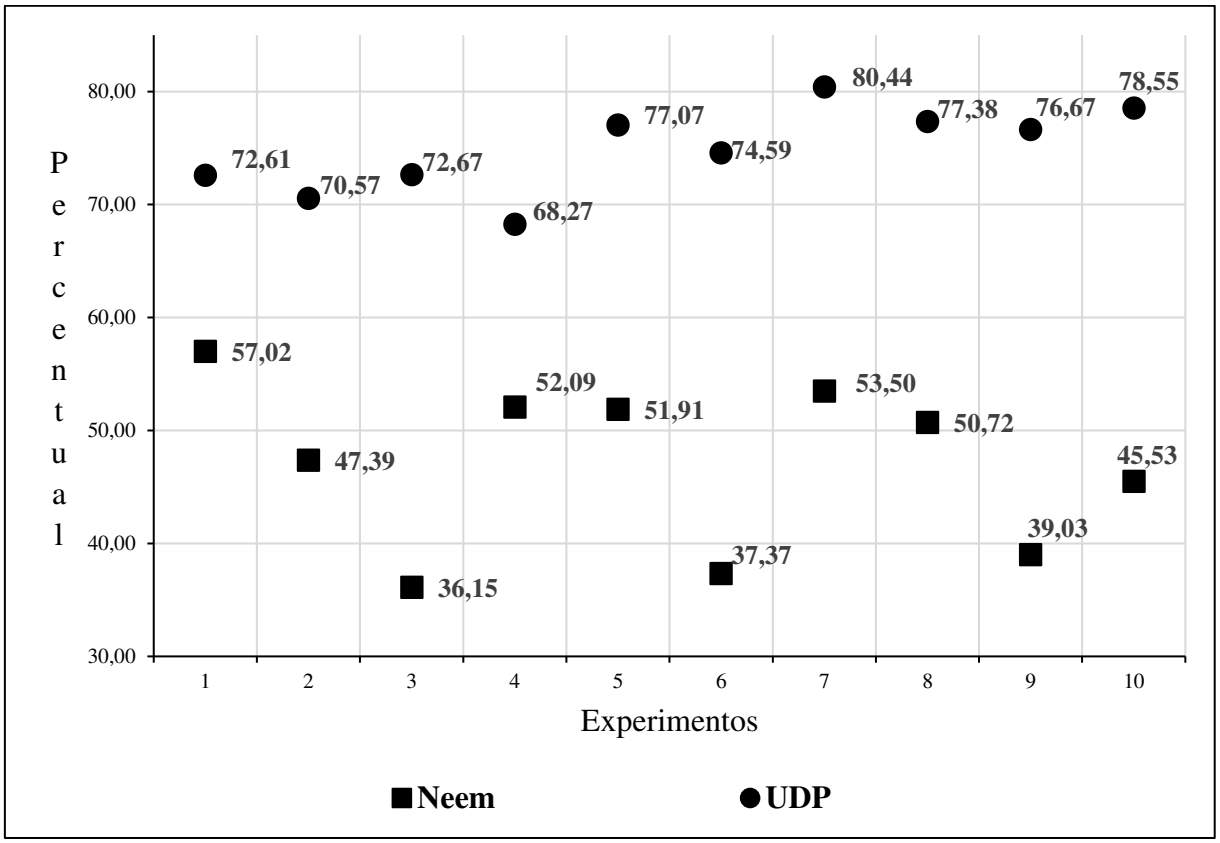


Fonte: Produção do próprio autor.

Para obter essa métrica foi calculada a média da quantidade de estações que receberam mensagens válidas. As diferenças obtidas são significativas, visto que em alguns casos experimentos rodando sobre o protocolo UDP atingiram até 11 estações a mais que o NEEM. Ao enviar mensagens para toda a rede sem parâmetros de controle da "infecção" como o *fanout*, as chances de mais estações serem atingidas são maiores, o que explica a vantagem significativa do UDP.

Na Figura 24 essa mesma métrica é mostrada levando em conta o percentual da cobertura da rede, considerando o número de 29 estações para a cobertura total. É possível observar que com a comunicação através do protocolo UDP, até 80% da rede receberam as mensagens válidas, enquanto no NEEM o máximo foi de 57,02%.

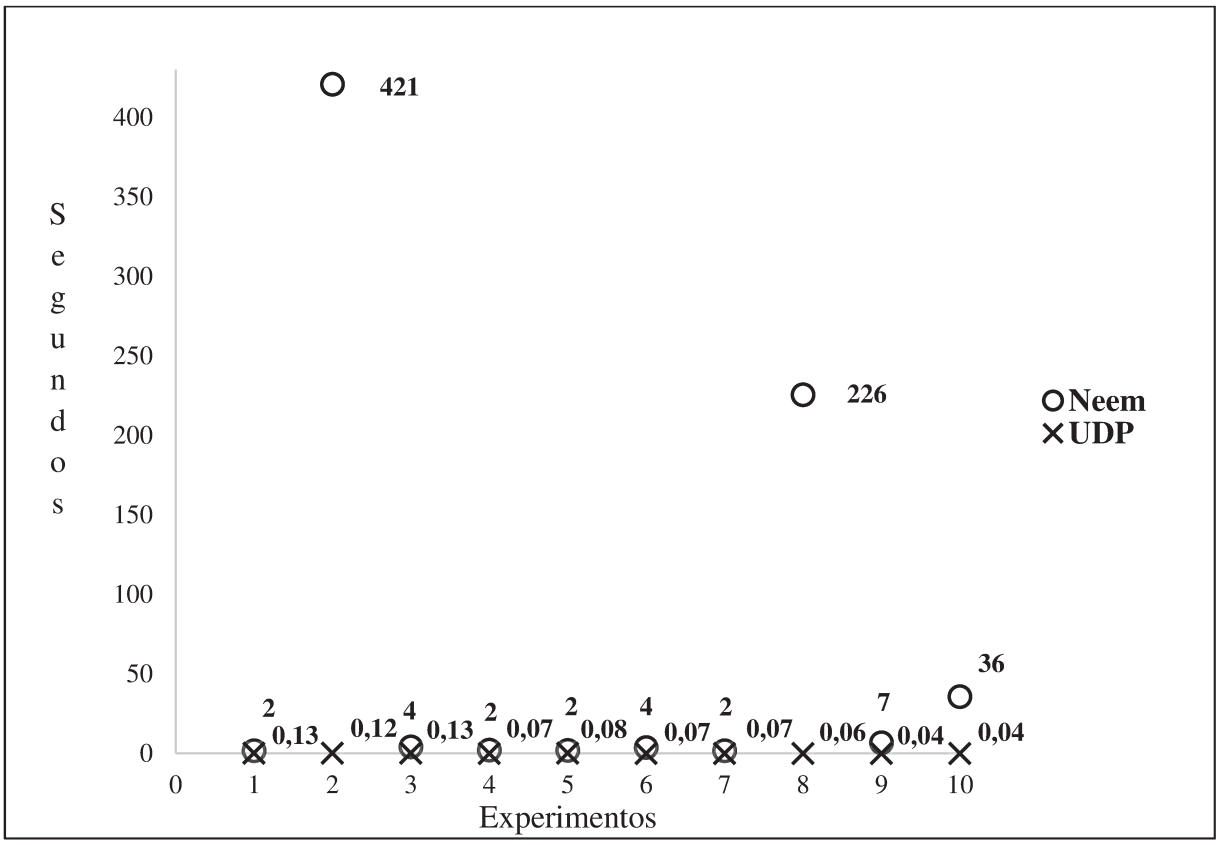
Figura 24- Média da cobertura total da rede com mensagens válidas



Fonte: Produção do próprio autor.

Outra métrica que trouxe resultados interessantes foi o tempo máximo que uma mensagem levou para ser entregue na rede, em média, considerando o último receptor desta. Nos testes com protocolo UDP, esse tempo foi abaixo de um segundo em todos os experimentos. Já com o NEEM, o tempo mínimo foi de 2 segundos, chegando a minutos nos experimentos 2 e 8. As causas da diferença atípica nesses casos em relação aos outros tempos não foram determinadas, sendo uma das possibilidades mensagens que estavam armazenadas em *buffer* para entrega posterior. Considerando o ambiente da aplicação, para a qual o tempo é um fator crítico, mensagens que são recebidas com muito atraso não atendem o objetivo da aplicação. A comparação entre os tempos pode ser vista na Figura 25.

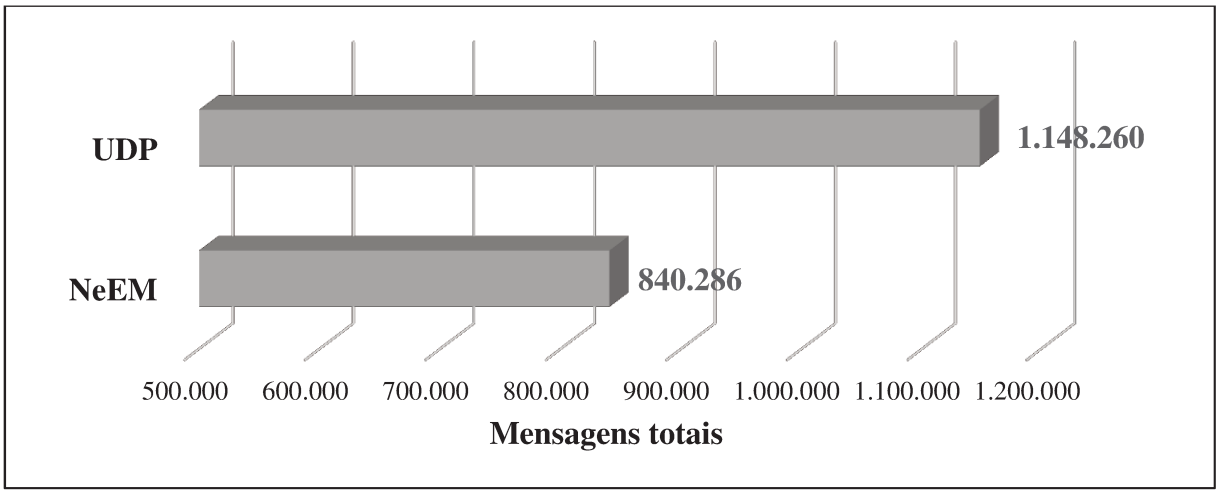
Figura 25- Média do tempo máximo de entrega



Fonte: Produção do próprio autor.

Os resultados apresentados até o momento mostraram as métricas obtidas em cada um dos experimentos. É interessante ainda mostrar as comparações da média geral de todos os experimentos. Dessa forma, ficaram ainda mais evidenciadas as diferenças de desempenho entre ambos protocolos. Na Figura 26, por exemplo, tem-se a comparação da média geral de mensagens totais.

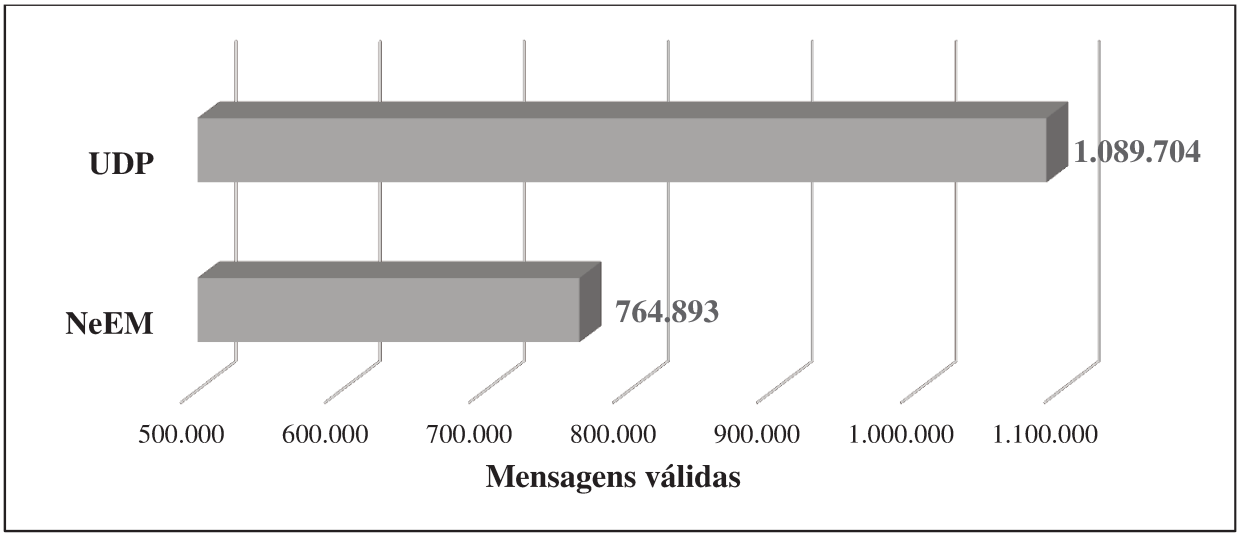
Figura 26- Média geral de mensagens totais por protocolo



Fonte: Produção do próprio autor.

Com o protocolo epidêmico NEEM, as estações recebem menos mensagens totais. Além disso, são apresentadas maiores taxas de mensagens descartadas conforme já mostrado na Figura 21, mesmo possuindo mecanismos internos para tentar mitigar problemas desta natureza. A situação não é diferente quando consideradas as mensagens válidas, onde prevalece a vantagem da comunicação através do protocolo UDP, como mostra a Figura 27.

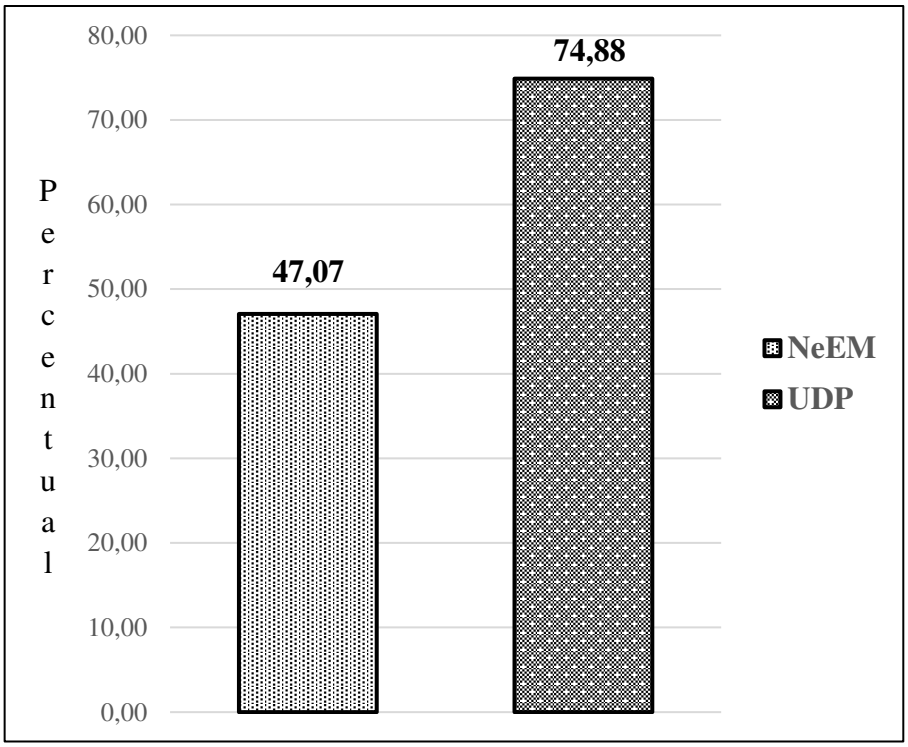
Figura 27- Média geral de mensagens válidas por protocolo



Fonte: Produção do próprio autor.

Ainda, uma observação interessante em relação a métrica de mensagens válidas é sua cobertura total na rede. A comunicação através do protocolo UDP apresenta maior taxa de cobertura total (média geral de 74,88%, enquanto com o NEEM, obteve-se 47,07%), como mostra a Figura 28.

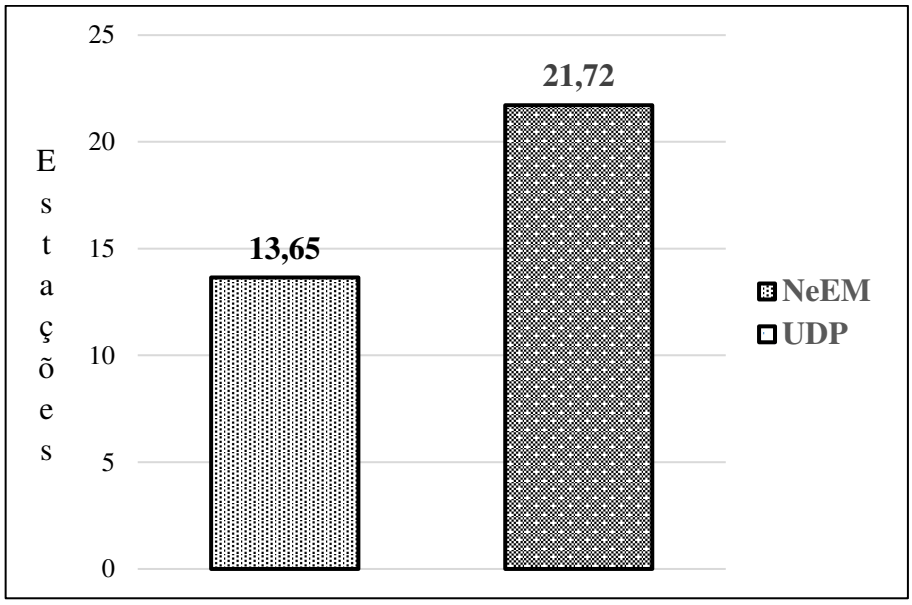
Figura 28 - Comparação da média geral de cobertura da rede



Fonte: Produção do próprio autor

A Figura 29 também mostra o comparativo de cobertura da rede, porém em relação à média geral da quantidade de estações que receberam as mensagens válidas. É possível observar de forma bastante clara a diferença na disseminação das mensagens na rede entre os dois protocolos: enquanto UDP atinge em média 21 estações, o NEEM atinge 13.

Figura 29- Comparação da média geral de estações atingidas



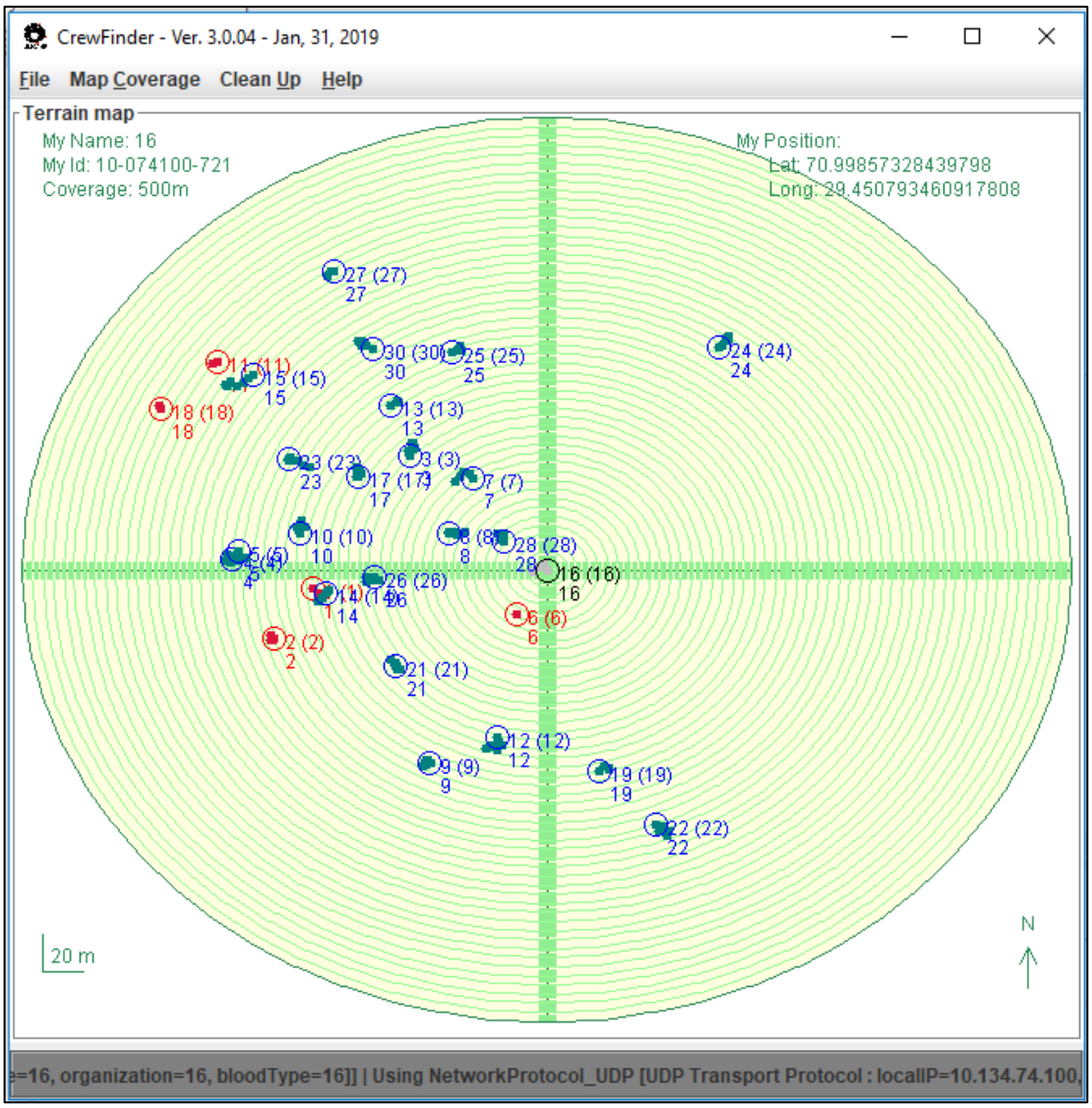
Fonte: Produção do próprio autor.

Tal diferença na cobertura da rede entre os protocolos pode ser justificada pelo fato do próprio NEEM dispor de parâmetros para controle da disseminação das mensagens, como o *fanout*. Como seu propósito é ser amigável a rede, tenta evitar sobrecargas e garantir a entrega de mensagens importantes, através da utilização do protocolo TCP. Por outro lado, na comunicação *broadcast* com o UDP, não existe mecanismos de gerenciamento para sobrecarga e nem a preocupação com a garantia das entregas, o que resulta em uma maior cobertura da rede.

A diferença era visível, inclusive, no momento da execução dos testes, quando alterações no tipo de "Tracks*"* eram percebidas por outras estações de forma praticamente instantânea quando testes eram realizados no UDP. Foi possível observar tal comportamento de forma padrão em todos os testes executados neste protocolo.

No NEEM, diversas situações diferentes puderam ser observadas. A título de exemplo, na Figuras 30, pode ser vista a tela da CrewFinder na estação 16 em um teste aleatório com a comunicação *broadcast* através do protocolo UDP.

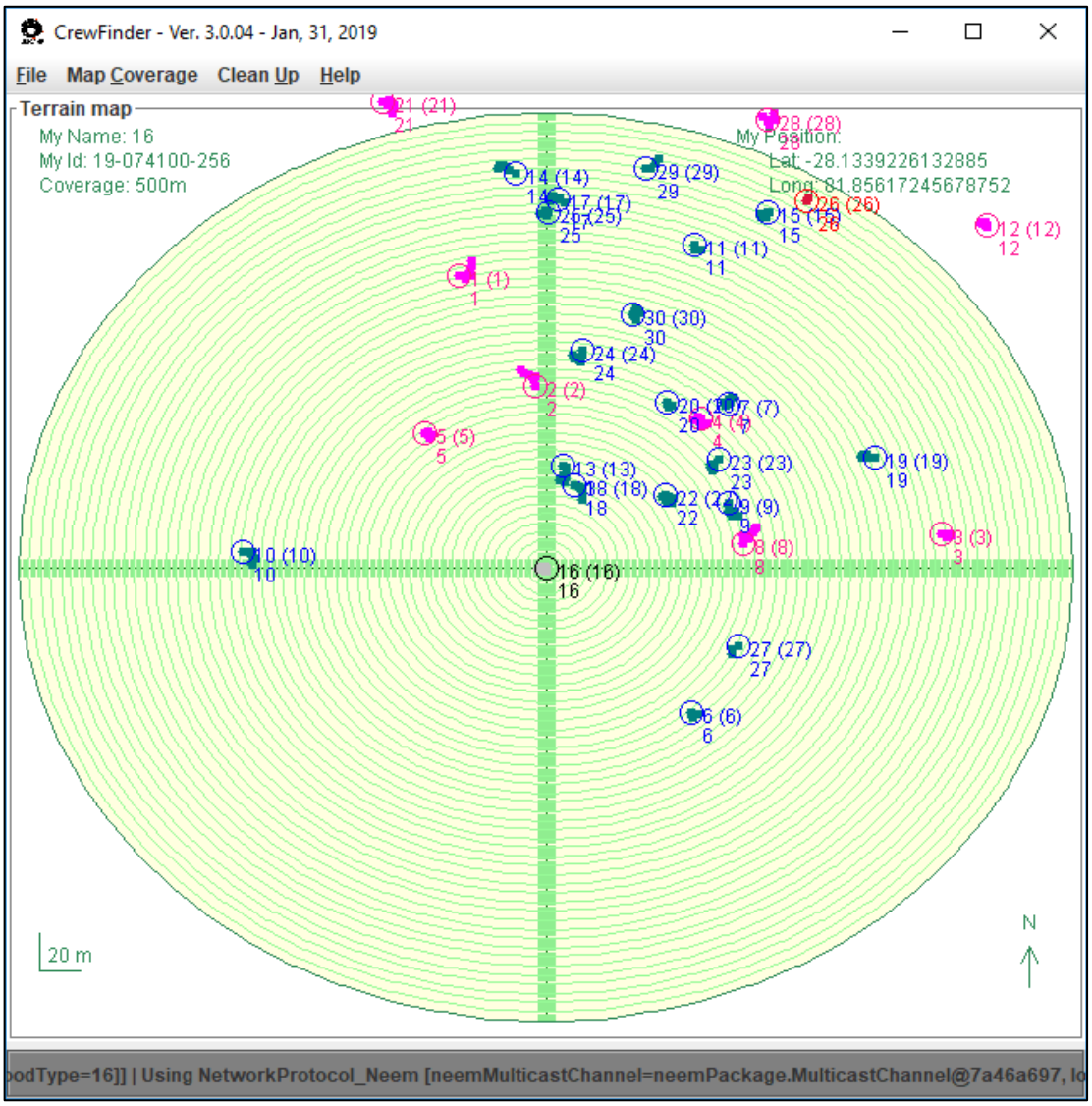
Figura 30- CrewFinder ao final de um experimento aleatório com UDP



Fonte: Produção do próprio autor.

O que pode ser observado é que as posições de todas as estações conhecidas pela aplicação continuam sendo recebidas ao final de uma hora de experimento – as de coloração vermelha indicam estações que não se movimentaram na última “Track” recebida. Enquanto na Figura 31, é mostrada a tela desta mesma estação 16 ao final de um experimento aleatório com o protocolo NEEM.

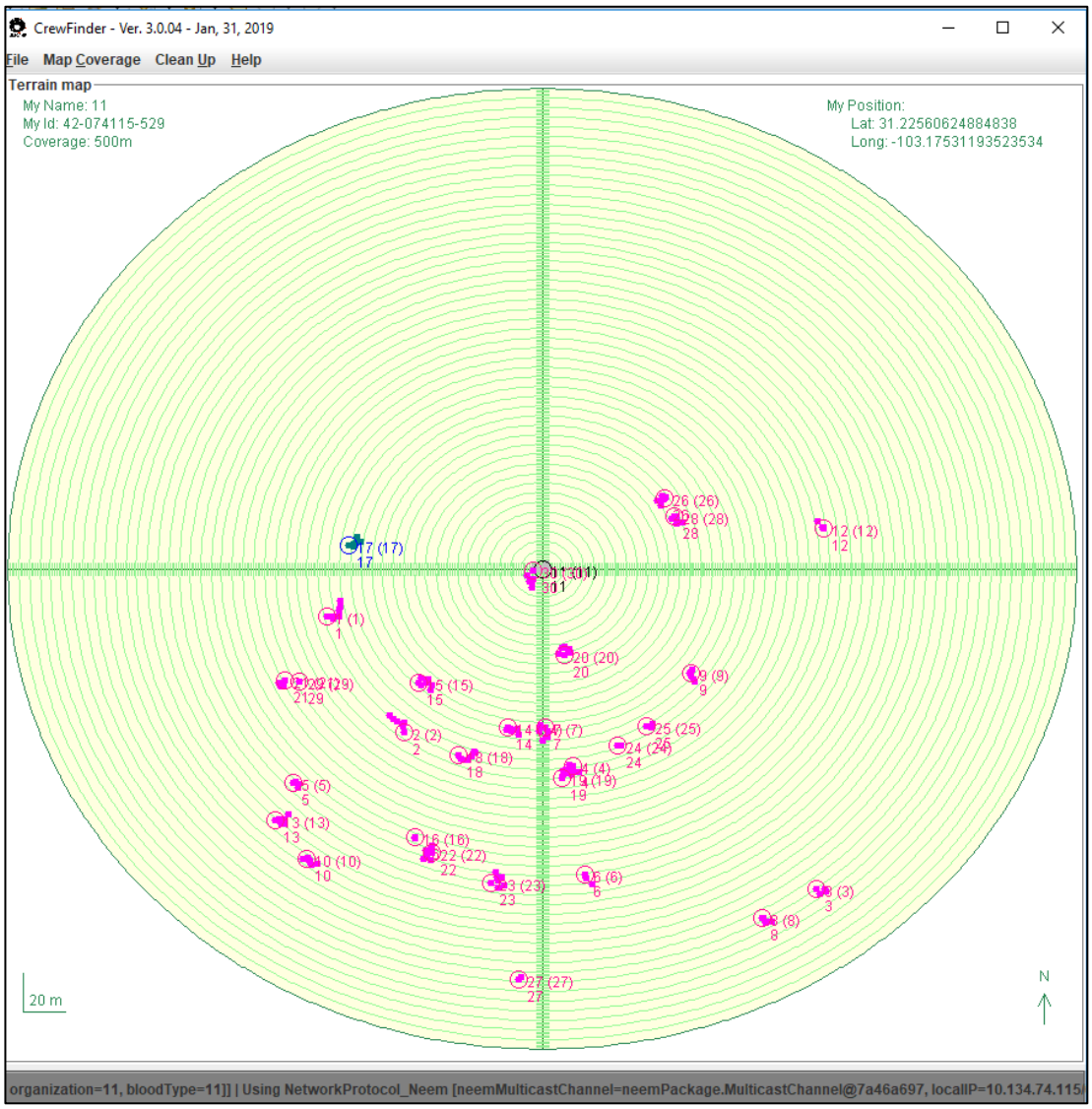
Figura 31- CrewFinder ao final de um experimento aleatório com NEEM



Fonte: Produção do próprio autor.

Diferente da Figura 30, posições de algumas estações conhecidas não são mais recebidas pela aplicação (tal reconhecimento é feito através da presença de trilhas na coloração rosa). Muito raramente situações como esta eram observadas nos experimentos com o *broadcast* UDP. Ainda, circunstâncias como a exibida na Figura 32 eram recorrentes em várias estações durante experimentos sobre o protocolo *multicast-gossip* NEEM.

Figura 32- CrewFinder ao final de um experimento aleatório com NEEM em outra estação



Fonte: Produção do próprio autor.

Neste caso, com exceção da estação 17 nenhuma outra posição estava sendo recebida ao final do experimento. Situações semelhantes a esta não foram observadas nos experimentos utilizando o protocolo UDP.

Diante de todos os resultados apresentados neste trabalho, comparações com [6] apresentam divergências. Em primeiro lugar, aquele cenário foi simulado. Além disso, uma única estação enviava mensagens para a rede através de uma aplicação construída sobre o pro-

tocolo ICMP, enquanto as demais apenas replicavam. Ainda, a aplicação empregada não continha todos os mecanismos do protocolo NEEM como o próprio gerenciamento de *buffer* por exemplo, limitando-se a apenas técnicas *gossip* para replicação das mensagens na rede. Tal configuração difere em todos os aspectos do cenário proposto no presente trabalho.

Dessa forma, divergências já eram esperadas, visto que este estudo empregou uma aplicação realista, construída sobre protocolos adequados, em um ambiente real que permitiu que as métricas fossem medidas e não somente simuladas, podendo assim fornecer resultados mais confiáveis para um ambiente real do que [6].

Já em sua concepção, o NEEM foi demonstrado com o jogo Microsoft Flight Simulator 2002. A descrição da metodologia sugere uma rede onde os jogadores estivessem conectados por *modems*[21] V.90 (56 kbps/33.6kbps), transmitindo 4 mensagens de atualização por segundo, cada qual com 60 *bytes* (*payload*). No entanto, a validação foi feita por simulação no NS2, tendo como entrada o padrão de tráfego esperado para a rede de jogadores, mas simulando entre 50 e 500 nós, sempre conectados com *uplink* e *downlink* de 56 kbps. Com efeito, o NEEM não foi validado em ambiente real em redes sem fio, sendo esta a grande contribuição deste trabalho.

Em linhas gerais, os resultados mostraram uma maior eficiência da comunicação UDP em todas as métricas avaliadas, ao mesmo tempo em que expôs as limitações do protocolo epidêmico NEEM neste ambiente específico e, por extensão, de outros protocolos epidêmicos que tenham características semelhantes. Este resultado não era o previsível, justamente por conta de o mecanismo epidêmico ser projetado para sincronizar sistemas distribuídos.

4.2 Notas gerais sobre os resultados

Inicialmente era esperada uma cobertura maior de rede com a comunicação epidêmica, já que esta foi concebida para tal finalidade, diferente do protocolo UDP. Porém, os resultados mostraram exatamente a situação oposta. Dessa forma, considerando uma aplicação

[21] Dispositivo eletrônico utilizado para conexão à Internet.

para o cenário proposto, a técnica *broadcast* entregou um serviço melhor do que o protocolo pensado com o objetivo de ser "amigo" da rede.

Embora exista um *overhead*[22] maior associado ao uso do TCP no caso do epidêmico NEEM, os pacotes trafegados eram de tamanhos semelhantes independente do protocolo utilizado, já que o conteúdo era o mesmo, podendo apresentar pequenas variações. Na realidade, inicialmente existia a expectativa que o uso do TCP fosse benéfico no sentido de garantir maiores taxas de entrega de mensagens, cujos resultados novamente mostraram o oposto.

Apesar de neste trabalho o NEEM ter sido avaliado em redes sem fio *ad hoc* reais, ainda assim não eram esperadas diferenças tão notórias de desempenho entre os protocolos. Considerando que o TCP é empregado na camada de transporte, presume-se que o protocolo seja independente do meio físico. Dessa forma, utilizá-lo em uma topologia de rede diferente de sua validação original não deveria ser um fator limitante. Além disso, sua concepção em ser amigável a rede e aliviar congestionamentos não o isenta de funcionar em redes não sobrecarregadas, ao mesmo tempo em que não garante seu funcionamento em uma rede totalmente congestionada [20].

Dessa forma, apesar de para esta aplicação específica o protocolo *multicast* epidêmico NEEM não ter sido superior em nenhuma métrica, o mesmo tem características interessantes que podem servir para outros contextos não explorados neste trabalho. A título de exemplo, em aplicações para longas distâncias ou mobilidades mais altas, as quais faça sentido algum mecanismo de garantia de transmissão ou armazenamento de mensagens em *buffer* para entregas posteriores, o uso do protocolo UDP pode ser apresentar limitações, já que não dispõe de tais recursos como o NEEM. Ainda, considerando um cenário com uma quantidade maior de estações enviando um número de pacotes suficiente para congestionar a rede, é um contexto bem provável do qual o emprego do TCP poderia trazer resultados mais satisfatórios. As avaliações de cenários como os exemplos citados anteriormente não foram consideradas no presente estudo.

[22] Gasto para executar determinada tarefa.

4.3 Considerações do capítulo

Este capítulo apresentou os resultados obtidos após as análises da comunicação desempenhada em cada experimento realizado. Para tanto, foi adotado um glossário de termos e apresentações através de figuras que proporcionam um melhor entendimento, com discussões complementando cada uma delas.

Além disso, foi demonstrado que comparações com trabalhos semelhantes, como se propunha inicialmente, mostraram-se prejudicadas diante da diferença entre as metodologias adotadas. Ainda, dada a notória e inesperada diferença de desempenho entre os protocolos, considerações foram feitas descrevendo que, apesar de ter características interessantes, o protocolo NEEM não se mostrou eficaz no contexto avaliado neste trabalho.

5. Conclusão

O presente trabalho analisou a comunicação *multicast-gossip* através do *Network Friendly Epidemic Multicast* - NEEM, comparando com a comunicação *broadcast* sobre o protocolo UDP em determinadas métricas, considerando o ambiente MANET – *mobile ad hoc networks*. Afim de atender os requisitos do cenário proposto, optou-se por desenvolver uma aplicação denominada CrewFinder para execução dos experimentos.

A integração da NeEM *library* na CrewFinder, que permitiu o envio de informações para a rede através da comunicação *multicast-gossip,* trouxe resultados interessantes a respeito do desempenho deste, já que o NEEM não havia sido projetado nem validado para MANETs em sua concepção original, assim como também não tinha sido validado em uma aplicação real neste ambiente sem o uso de simuladores.

O protocolo *multicast-gossip* NEEM, ao se deparar com a inundação de mensagens redundantes na rede, utiliza-se de seus gerenciamentos para tentar melhorar a comunicação e garantir as entregas dos pacotes através do uso do protocolo TCP, que dispõe de técnicas para prover tal garantia. Porém, tais mecanismos utilizados pelo protocolo NEEM, além de entregar menos mensagens, acabam fazendo com que a entrega seja mais demorada, resultando em mensagens antigas e consequentemente descartadas pela aplicação. Ainda, em termos de "infecção" na rede, ou seja, em número de estações atingidas pelas mensagens (principal característica pelas quais a comunicação epidêmica *multicast-gossip* são conhecidas), o uso simples do protocolo UDP em *broadcast* também se mostra mais eficiente dentro das condições analisadas.

O diferencial do NEEM em relação aos outros protocolos epidêmicos, com sua proposta em ser amigável a rede através do uso do TCP e descarte de mensagens em situações de sobrecarga na rede, que a princípio o tornara atrativo para MANETs por conta das limitações presentes neste ambiente, foi possivelmente a própria causa de seu desempenho ter sido inferior a comunicação *broadcast* com UDP, já que no segundo não existe preocupação nenhuma com garantia de entrega, por exemplo.

Assim sendo, foi possível concluir que, em uma aplicação para MANETs com baixa mobilidade, o uso da comunicação *broadcast* através do protocolo UDP mostrou-se mais eficaz

em todas métricas avaliadas, em especial nas mais sensíveis para a aplicação proposta, como tempo máximo de entrega e cobertura geral da rede, nas quais as diferenças apresentadas pelas duas formas de comunicação foram bastante significativas.

Esta pesquisa atingiu seus objetivos, com as seguintes contribuições:

a. Análise de um protocolo epidêmico no ambiente das MANETs suportando uma aplicação realista;

b. A demonstração de que no cenário proposto, a comunicação UDP por *broadcast* é superior em todas as métricas consideradas;

c. A criação de uma aplicação com potencial para testar facilmente protocolos no cenário examinado;

d. Um protótipo e prova de conceito de um aplicativo possivelmente útil em situações de emergência;

e. A constatação da divergência de terminologia entre a comunidade de redes de computadores e aquela envolvida na comunicação epidêmica;

f. A indicação do funcionamento da implementação da biblioteca NEEM *library* e viabilidade de sua integração a programas que se interessarem pelo mecanismo epidêmico.

São sugestões de trabalhos futuros as seguintes:

a. O teste do protocolo epidêmico em MANETs em diferentes domínios;

b. A análise de sensibilidade dos parâmetros do NEEM;

c. O desenvolvimento de protocolos com desempenho superior ao *broadcast* UDP para o cenário de atendimento a emergências;

d. A investigação de mecanismos de segurança que possam ser adequados ao cenário considerado;

e. A inclusão de mecanismos de economia de energia no nível de protocolo, visto que a duração das baterias é fator crítico;

f. O desenvolvimento de metodologias de avaliação de desempenho de protocolos epidêmicos em MANETs com mobilidade real.

Este capítulo apresentou as conclusões do presente estudo, fundamentadas nos resultados encontrados. Ainda, foram indicados trabalhos futuros que abrem diversas possibilidades para novas pesquisas, demonstrando que o estudo desenvolvido foi bastante satisfatório ao atingir seu principal objetivo, além de deixar importantes contribuições.

6. Referências

[1] TORRES, G. **Redes de computadores**. 2. ed. Rio de Janeiro: Nova Terra Editora, 2014.

[2] FOROUZAN, A. B. **Comunicação de dados e Redes de Computadores**. 4. ed. Porto Alegre: AMGH Editora, 2008.

[3] DAVIE, B.S.; PETERSON, L.L. **Redes de computadores: uma abordagem de sistemas**. 5. ed. Rio de Janeiro: Elsevier Editora, 2013.

[4] ROSS, J. **O livro do wireless: um guia definitivo para wi-fi e redes sem fio**. 2. ed. Rio de Janeiro: Alta Books, 2009.

[5] BANG, O.A.; RAMTEKE, P. **MANET: History, Challenges and Applications**. International Journal of Application or Innovation in Engineering & Management, v. 2, Issue 9, September 2013.

[6] LEME, E. **Análise de falhas em comunicação multicast-gossip no ambiente MANET**. 118 f. Dissertação (Mestrado em Tecnologia) – Faculdade de Tecnologia – Universidade de Campinas, Limeira. 2016.

[7] BRIGNONI, G. **Estudo de Protocolos de Roteamento em Redes Ad Hoc.** 73 f. Monografia (Bacharel em ciência da computação) – Universidade Federal de Santa Catarina, Florianópolis. 2005.

[8] FRIEDMAN, R.; GAVIDIA, D.; RODRIGUES, L.; VIANA, A.C.; VOULGARIS, S. **Gossiping on MANETs: The Beauty and the Beast**. ACM SIGOPS Oper. Syst. Rev., v. 41, p. 67–74, 2007.

[9] ENDO, P. T.; JULIÃO, E. C. B, SILVA, E. R. **Uma análise comparativa entre protocolos de comunicação autonômicos baseados em *gossip*.** Faculdade de Ciências e Tecnologia de Caruaru – Universidade de Pernambuco – Caruaru. Revista da Escola Regional de informática. v.2, p.105-113, 2013.

[10] HADDADOU, N. **Vehicular ad hoc networks: towards efficient, collaborativeand reliable data dissemination**. Tese (Doutorado em Matemática e ciências e Tecnologia da informação e da comunicação) - École Doctorale MSTIC - Universidade de Paris, Paris. 2014.

[11] ALMEIDA, V. D. D. **Análise de Desempenho de Protocolos de Roteamento AD HOC e DTN em Redes de Emergência**. 90 f. Dissertação (Mestrado em ciência da computação). Instituto de Ciências Exatas da Universidade Federal de Minas Gerais, Belo Horizonte, 2011.

[12] MANOJ, B.S.; BAKER, A.H. **Communication challenges in emergency response**. Communications of the ACM - Emergency response information systems: emerging trends and technologies,v. 50 Issue 3,. New York, USA, 2007.

[13] OLIVEIRA, T. R., OLIVEIRA, S., NOGUEIRA, J. M. **Um Modelo de Gerenciamento de Segurança Adaptativo para Redes de Emergência.** In: Anais XXVIII simpósio brasileiro de redes de computadores e sistemas distribuídos. Minas Gerais, Belo Horizonte. P.31-44.

[14] KUROSE, J.; ROSS, K. **Redes de computadores e a internet: uma abordagem top down.** 6. ed. São Paulo: Pearson Education do Brasil, 2014.

[15] IEEE 802.11, 1999 Ed. (Iso/Iec 8802-11: 1999), **IEEE Standards for Information Technology — Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Network — Specific Requirements** — Part 11: Wireless Lan Medium Access Control (Mac) And Physical Layer (Phy) Specifications. 1999.

[16] GOYAL, P.; PARMAR, V.; RISHI, R. **MANET: Vulnerabilities, Challenges, Attacks, Aplication**. International Journal of Computational Engineering & Management, v. 11, p. 2230-7893, 2011.

[17] RAZA, N. et al. **Mobile Ad-Hoc Networks Applications and Its Challenges.** Communications and Network, v.8, p. 131-136, 2016.

[18] AHMED, D. E. M.; Khalifa, O. O. **An Overview of MANETs: Applications, Characteristics, Challenges and Recent Issues**. International Journal of Engineering and Advanced Technology, v.6, p. 128-133, 2017.

[19] DEMERS, A; et al. **A. Epidemic Algorithms for Replicated Database Maintenance.** In: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing. p. 1-12. doi>10.1145/41840.41841. 1987.

[20] PEREIRA J.; RODRIGUES L.; MONTEIRO M. J.; OLIVEIRA R.; KERMARREC **A. NEEM: Network-Friendly Epidemic Multicast**. IEEE 22 International Symposium on Reliable Distributed Systems, p. 15-24, Florence, Italy, 2003

[21] LEITÃO, J. A. **Gossip-Based Broadcast Protocols**. 92f. Dissertação (Mestrado em Engenharia de Informática) - Universidade de Lisboa, Lisboa - Portugal, 2007.

[22] FREY, D.; GUERRAOUIL, R.; KERMARREC, A.; KOLDEHOFE, B.; MOGENSEN, M.; MONOD, M.; QUÉMA, V.; **Heterogeneous Gossip.** In: International Conference on Middleware. Nova Iorque, 2009. p. 42-61.

[23] DESHPANDE, M.; XING, B.; LAZARDIS, I.; VENKATASUBRAMANIAN, N.; MEHROTA, S. **Crew: A gossip-based flash-dissemination system**. In ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems, 45, IEEE Computer Society, Washington, DC, USA, 2006.

[24] COMER, D. E**. Interligação de redes com TCP/IP princípios, protocolos e arquitetura.** 6. ed. Rio de Janeiro: Elsevier, 2015.

[25] MONTRESOR, A. **Intelligent Gossip. In: Intelligent Distributed Computing, Systems and Applications**. Springer Berlin Heidelberg, 2008. p. 3-10.

[26] JELASITY, M. **Gossip-based Protocols for Large-scale Distributed Systems.** 168 f. Dissertação (Doutorado em ciência). University of Szeged - Szeged, 2013.

[27] BIRMAN, K. **The Promise, and Limitations, of Gossip Protocols.** SIGOPS Open. Syst. Rev., v. 41, p.8–13, 2007.

[28] GOLDING, R.; TAYLOR, K. **Group membership in the epidemic style.** Technical Report UCSC-CRL-92-13, UC Santa Cruz, Dept. of Computer Science, May 1992. 12f.

[29] PEREIRA, J.O. **NEEM Library chat.** [mensagem pessoal]. Mensagem recebida por <lucas.bonetti@ifsp.edu.br> em 01 out. 2018.

[30] HAYDEN, M.; BIRMAN, K. **Probabilistic broadcast**. Technical Report TR96-1606, Cornell University, Computer Science, 1996. 15 f.

[31] EUGSTER, P. T.; GUERRAOUI, R.; KERMARREC, A. M.; MASSOULIÉ, L. **Epidemic information dissemination in distributed system**. Computer. v. 37, p. 60-67, 2004.

[32] OLIVEIRA, G. P. A. **Monitoramento e Análise visual de tráfego de dados de aplicações baseadas em protocolos epidêmicos**. 83f. Monografia (Bacharel em Ciências da Computação) - Instituto de Informática - Universidade Federal do Rio Grande do Sul. Porto Alegre, 2012.

[33] PEREIRA, J.O.; SANTOS, P. **NeEM: Network-friendly Epidemic Multicast (NeEM library**). Disponível em: < http://neem.sourceforge.net/software.html >. Acesso em 13/06/2017.

[34] Corpo de bombeiros: Polícia Militar do estado de São Paulo. **Bombeiros em números**. Disponível em: < http://www.corpodebombeiros.sp.gov.br/ >. Acesso em 19/11/2018.

[35] WILGES, P., CECHIN, S. L.,WEBER, T. S., MORAES, R. **A Distributed Presence Service over Epidemic Multicast.** Journal of Applied Computing Research v.2, p. 50-59, 2012.

[36] PETRI, M., KANIESS, J., PAIRPINELLI, R. S. **Algoritmo genético para escalonamento de recursos em descoberta de serviços para MANETs.** In: XLIX Simpósio Brasileiro de Pesquisa Operacional. Santa Catarina, Blumenau.

[37] MOTA, V. F. S., SILVA, T. H., NOGUEIRA, J. M. S. **Introduzindo tolerância a interrupção em Redes Ad Hoc Móveis para Cenários de Emergência.** In: Anais XXVII Simpósio brasileiro de redes de computadores e sistemas distribuídos. Pernambuco, Recife. P.671-684.

7. Apêndices

Apêndice A – Estrutura do arquivo de *log*

Exemplo de três linhas aleatórias de um arquivo de *log* gerado por uma estação executando a aplicação CrewFinder (os separadores dos campos são marcados por $):

07-229083-038 $ 2019/02/06 17:26:49 $ Just received $ RECEIVED $ 10-240249-481 $ MYTRACK $ 1549481189672 $ 7 $ 169, 1549481187660, 10.773988557040427, -3.6010884621287635 $ 168, 1549481185647, 10.773988557040427, -3.6010884621287635 $ 167, 1549481183635, 10.773988557040427, -3.6010884621287635 $ 166, 1549481181623, 10.773988557040427, -3.6010884621287635 $ 165, 1549481179610, 12.36480613671818, -4.19254480231513 $ 164, 1549481177598, 13.189010050754524, -7.550838774875764 $ 163, 1549481175585, 10.067156929340532, -9.966329652264157 $ $

07-229083-038 $ 2019/02/06 17:46:51 $ Just received $ RECEIVED $ 26-046214-394 $ MYTRACK $ 1549482412304 $ 7 $ 773, 1549482410292, 23.708851515828535, -79.92199021665192 $ 772, 1549482408279, 21.639985330530777, -81.12050772247896 $ 771, 1549482406267, 20.791169751282922, -83.85180747327706 $ 770, 1549482404255, 23.51400396328524, -84.26333381780563 $ 769, 1549482402242, 22.611636919533286, -81.00962179972021 $ 768, 1549482400230, 19.25333125438151, -77.66020371035816 $ 767, 1549482398217, 21.54459231679052, -75.02396576676286 $ $

07-229083-038 $ 2019/02/06 17:57:13 $ Just received $ RECEIVED $ 68-218110-415 $ MYTRACK $ 1549483037227 $ 7 $ 1021, 1549483035215, -70.23778244062592, -74.90166109444445 $ 1020, 1549483033203, -71.97865156162815, -74.65698529691409 $ 1019, 1549483031190, -72.41221569678868, -76.55477757827859 $ 1018, 1549483029178, -71.99514827035588, -78.64841941395208 $ 1017, 1549483027165, -75.32125872685677, -76.34378090542191 $ 1016, 1549483025153, -75.88684791600346, -73.41781379836593 $ 1015, 1549483023140, -78.04814226423383, -72.98937971757464 $ $

Apêndice B – Comandos SQL utilizados

Os comandos utilizados para criação da tabela no banco de dados são representados na Figura 33.

Figura 33- Comandos para criação da tabela no banco de dados

```
CREATE TABLE `experiments` (
  `stationID` varchar(20) CHARACTER SET latin1 NOT NULL,
  `date` datetime NOT NULL,
  `process` varchar(20) CHARACTER SET latin1 NOT NULL,
  `action` varchar(20) CHARACTER SET latin1 NOT NULL,
  `senderID` varchar(20) CHARACTER SET latin1 NOT NULL,
  `type` varchar(20) CHARACTER SET latin1 NOT NULL,
  `lastupdate` bigint(20) NOT NULL,
  `tracks` int(11) NOT NULL,
  `track1` varchar(100) CHARACTER SET latin1 DEFAULT NULL,
  `track2` varchar(100) CHARACTER SET latin1 DEFAULT NULL,
  `track3` varchar(100) CHARACTER SET latin1 DEFAULT NULL,
  `track4` varchar(100) CHARACTER SET latin1 DEFAULT NULL,
  `track5` varchar(100) CHARACTER SET latin1 DEFAULT NULL,
  `track6` varchar(100) CHARACTER SET latin1 DEFAULT NULL,
  `track7` varchar(100) CHARACTER SET latin1 DEFAULT NULL,
  `register` bigint(20) NOT NULL AUTO_INCREMENT,
  `protocol` varchar(4) CHARACTER SET latin1 DEFAULT NULL,
  `experimentID` int(2) DEFAULT NULL,
  PRIMARY KEY (`register`)
) ENGINE=InnoDB AUTO_INCREMENT=0 DEFAULT CHARSET=utf8;
```

Fonte: Produção do próprio autor.

O comando utilizado para importar o arquivo de texto de *log* diretamente no banco de dados pode ser visto na Figura 34.

Figura 34- Comando de importação dos *logs*

```
LOAD DATA LOCAL INFILE '/var/lib/mysql-files/testes/neem/1/LogFileOf_00-074112-716.txt' INTO TABLE experiments FIELDS TERMINATED BY '$';
```

Fonte: Produção do próprio autor.

A consulta realizada para obter a quantidade de mensagens fora de alcance em um determinado teste é mostrada na Figura 35.

Figura 35- Consulta para obter métricas de mensagens fora de alcance

```
select e.senderID, e.lastupdate from experiments as e where e.protocol='udp' and e.experimentID=10 and e.process='out of range';
```

Fonte: Produção do próprio autor.

A consulta realizada para obter a quantidade de mensagens totais em um determinado teste é mostrada na Figura 36.

Figura 36- Consulta para obter métricas de mensagens totais

```
select (recebida.total - outofrange.total) as total from
(select count(e.senderID) as total from experiments e where e.protocol='neem' and e.experimentID=1 and action='RECEIVED') as recebida,
(select count(a.senderID) as total from experiments a where a.protocol='neem' and a.experimentID=1 and process='out of range') as outofrange;
```

Fonte: Produção do próprio autor.

A consulta realizada para obter a quantidade de mensagens descartadas em um determinado teste é mostrada na Figura 37.

Figura 37- Consulta para obter métricas de mensagens descartadas

```
select e.senderID, e.lastupdate from experiments as e where e.protocol='neem' and e.experimentID=1 and e.process='own track' or e.protocol='neem' and e.
experimentID='1' and e.process='old or duplicated';
```

Fonte: Produção do próprio autor.

A consulta realizada para obter a quantidade de mensagens válidas em um determinado teste é mostrada na Figura 38.

Figura 38- Consulta para obter métricas de mensagens válidas

```
select (recebida.total - descarte.total) as total from
(select count(e.senderID) as total from experiments e where e.protocol='neem' and e.experimentID=1 and action='RECEIVED') as recebida,
(select count(a.senderID) as total from experiments a where a.protocol='neem' and a.experimentID=1 and action='DISCARDED') as descarte;
```

Fonte: Produção do próprio autor.

A consulta realizada para obter a quantidade de estações que receberam mensagens válidas em um determinado teste é mostrada na Figura 39.

Figura 39- Consulta para obter métricas das estações atingidas por mensagens válidas

```
select d.senderID, d.lastupdate, count(d.stationID) as nós from (
select distinct e.stationID, e.senderID, e.lastupdate, e.action from experiments as e where e.protocol='neem' and e.experimentID=1 and action='RECEIVED'
and e.stationID != e.senderID) as d group by d.senderID, d.lastupdate;
```

Fonte: Produção do próprio autor.

A consulta realizada para obter as mensagens totais durante um determinado experimento é representada na Figura 40.

Figura 40- Consulta para obter métricas de mensagens totais

```
select e.senderID, e.lastupdate from experiments as e where e.protocol='udp' and e.experimentID=1 and action='RECEIVED';
```

Fonte: Produção do próprio autor.

A consulta realizada para obter o tempo máximo que cada mensagem levou para ser entregue na rede em um determinado experimento pode ser vista na Figura 41.

Figura 41- Consulta para obter métricas de tempo máximo de recebimento das mensagens

```
select e.senderID, e.lastupdate as criada, UNIX_TIMESTAMP(max(e.date)) as recebida from experiments as e where e.protocol='udp' and e.experimentID=1 and
action='RECEIVED' and e.stationID != e.senderID group by e.senderID, e.lastupdate;
```

Fonte: Produção do próprio autor.

8. Anexos

Anexo A – Dados Oficiais do Corpo de Bombeiros do Estado de São Paulo

Figura 42- Números oficiais dos Bombeiros de São Paulo

Fonte: Retirada de [34].