Universidade do Minho
Escola de Engenharia

Luís Paulo de Sousa Ramalho

Avaliação de desempenho de redes oportunistas

Julho de 2016

Universidade do Minho
Escola de Engenharia

Luís Paulo de Sousa Ramalho

Avaliação de desempenho de redes oportunistas

Dissertação de Mestrado
Mestrado em Engenharia de Redes e Serviços Telemáticos

Trabalho efectuado sob a orientação do(s)
Professor Doutor Adriano Moreira
Professor Doutor Filipe Meneses

Julho de 2016

AGRADECIMENTOS

Ao meu orientador Professor Doutor Adriano Moreira por ter aceite a orientação da minha dissertação e por todo o apoio, paciência, disponibilidade e dedicação demonstrada ao longo de todo o desenvolvimento da mesma, agradecendo igualmente todas as sugestões e ensinamentos transmitidos ao longo deste tempo.

Aos meus pais agradeço toda a dedicação e preocupação constantes e sobretudo todos os esforços realizados para que o melhor me fosse proporcionado.

À minha irmã por todo o apoio, carinho e amizade.

À minha namorada pelo imenso apoio e motivação demonstrados ao longo do tempo e principalmente pelo seu amor e paciência.

A todos os meus colegas académicos que ao longo desta fase me ajudaram e apoiaram, o meu obrigado.

Agradeço igualmente a todos os meus amigos pela sua amizade e pelo seu apoio e ajuda que demonstraram sempre que necessitei.

A todos aqueles que de alguma forma contribuíram para a realização desta dissertação, o meu muito obrigado.

RESUMO

Nos últimos anos, a grande utilização de dispositivos móveis tem levado a um considerável aumento de interesse da comunidade de investigação e da indústria na área das redes oportunistas. Nestas redes, os dispositivos móveis aproveitam a sua mobilidade para trocarem dados entre si. Uma vez que estes dispositivos são principalmente transportados por humanos, surgem oportunidades de contacto aquando da proximidade física entre eles, permitindo essa troca de dados. É por esta razão que os investigadores têm vindo a explorar a mobilidade humana, aplicando-a ao estudo destas redes.

Neste contexto surge o objetivo principal deste trabalho que visa explorar a disponibilidade de dados reais sobre o movimento de pessoas para avaliar o desempenho de redes oportunistas, considerando que a análise de *traces* Wi-Fi pode fornecer informações importantes sobre o assunto, em particular, para estimar o atraso e a largura de banda. Entre outros, um dos grandes desafios a abordar é a grande quantidade de dados a analisar (*big data*), onde serão consideradas técnicas para analisar esses grandes volumes de dados de forma eficiente.

Nesse sentido foram desenvolvidos algoritmos que permitissem avaliar o desempenho destas redes. Um primeiro capaz de gerar *logs* RADIUS sintéticos num ambiente controlado, onde se pode configurar a simulação com os valores que se pretendam. A sua validação possibilitou constatar a eficiência do mesmo, permitindo gerar corretamente uma grande quantidade de registos sintéticos e num curto espaço de tempo. Um segundo algoritmo que possibilita extrair encontros de uma lista de registos preexistente. A validação deste algoritmo também permitiu verificar que é possível extrair uma grande quantidade de encontros de forma simples e rápida. Por fim, um terceiro algoritmo que visa o encaminhamento epidémico de mensagens a partir de um conjunto de encontros preexistente e através da utilização de duas diferentes estratégias de transmissão de mensagens.

A partir dos resultados alcançados nas experiências realizadas com o terceiro algoritmo, foi possível constatar-se que de facto as DTNs caracterizam-se pelas suas baixas taxas de entrega. Apesar disso, face ao protocolo aplicado nas experiências realizadas, esperavam-se taxas mais elevadas. Foi possível igualmente constatar-se que a taxa de entrega a alcançar é influenciada pela estratégia de transmissão de mensagens que se adote, uma vez que foi sempre superior quando se utilizou a segunda estratégia. Por outro lado, através dos resultados obtidos foi também possível concluir que a variação do tamanho da mensagem não se reflete na taxa de entrega alcançada.

Os resultados também confirmaram o facto do protocolo epidémico requerer uma enorme quantidade de recursos, dada à excessiva quantidade de transmissões que efetua e ao considerável espaço de armazenamento que as stations necessitam ter. Assim, dado ao facto das stations serem nós móveis e possuírem baterias relativamente limitadas, é de prever que tendam a ficar sem energia passado algum tempo.

Ao nível do atraso, os resultados mostraram que os valores médios e máximos dependeram sempre da estratégia de transmissão de mensagens que se utilizou. Por outro lado, não se verificou uma dependência em relação ao tamanho das mensagens geradas na rede, uma vez que não existiu uma tendência visível nos resultados ao variar o tamanho das mesmas.

PALAVRAS-CHAVE: redes oportunistas, dispositivos móveis, oportunidades de contacto, *big data*

ABSTRACT

Nowadays with the dissemination of mobile devices, the interest from the research community and the industry in opportunistic networks has increased significantly. In these networks, mobile devices take advantage of their mobility to exchange data with each other. Since these devices are mainly carried by humans, contact opportunities arises during physical proximity among them, enabling data exchange. It is for this reason that researchers have been exploring human mobility applied to the study of these networks.

In this context arises the main objective of this study which aims to explore the availability of real data about the movement of people to assess the performance of these networks, considering that the analysis of Wi-Fi data traces can provide significant information about this, in particular can be use to estimate delay and bandwidth. Among others, one of the major challenges to address is the large amount of data to be analyzed (big data), where will be considered techniques to analyze these large volumes of data efficiently.

In this sense, algorithms were developed to allow to assess the performance of these networks. A first able to generate synthetic RADIUS records in a controlled environment, where one can configure the simulation with values desired. Their validation allowed to verify the efficiency of the same, generating a large amount of synthetic records correctly and in a short time. A second algorithm that allows extracting encounters from a pre-existing list of records. The validation of this algorithm also allowed to verify that it is possible extract a large amount of encounters simply and quickly. Finally, a third algorithm that aims the epidemic routing of messages from a set of pre-existing encounters using two different message transmission strategies.

From the results achieved in experiments with the third algorithm, it was possible to verify that in fact DTNs are characterized by their low delivery rates. Despite this, given the protocol applied in the experiments, were expected higher rates. It was also possible to verify that the delivery rate achieved is influenced by the message transmission strategy adopted, since it was always higher when the second strategy was used. On other hand, based on the obtained results it was also possible concluded that the variation of the size of the message is not reflected in the delivery rate achieved.

The results also confirmed that the epidemic protocol require a huge amount of resources due to the excessive amount of transmissions that performs and significant storage space that stations need to have. Thus, given the fact that the stations are mobile nodes and having relatively small batteries, it is expected that they will tend to run out of energy spent some time.

At the delay level, the results showed that the average and maximum values always depended on the message transmission strategy that was used. On the other hand, not there was a dependence on the size of messages generated in the network, because there has been no noticeable tendency in the results when varying their size.

KEYWORDS: opportunistic networks, mobile devices, contact opportunities, big data

ÍNDICE GERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ACRÓNIMOS E ABREVIATURAS

AP	*Access Point*
CAR	*Context-Aware Routing*
CTG	*Connectivity Traces*
DSDV	*Destination-Sequenced Distance Vector*
IDE	*Integrated Development Environment*
JDBC	*Java Database Connectivity*
MANET	*Mobile Ad Hoc Network*
OO	*Orientado a Objetos*
OppNets	*Opportunistic Networks*
PDA	*Personal Digital Assistant*
POO	*Programação Orientada a Objetos*
RADIUS	*Remote Authentication Dial In User Service*
SNMP	*Simple Network Management Protocol*
SREP	*Social Relationship Enhanced Predicable*
SSAR	*Social Selfishness Aware Routing*

CAPÍTULO 1
INTRODUÇÃO

1.1 Enquadramento

A disseminação em larga escala de pequenos dispositivos portáteis, tais como sensores sem fios, *smartphones*, *tablets*, e mesmo dispositivos instalados em automóveis, tem levado a um aumento de interesse da comunidade de investigação nesta área e a criar as condições para o surgimento de um novo tipo de redes de comunicações designadas por redes oportunistas, ou redes tolerantes a atrasos.

Nestas redes, os dispositivos aproveitam a proximidade física entre si para trocarem dados, mesmo que não exista uma infraestrutura de telecomunicações presente no local. O principal objetivo é beneficiar da mobilidade desses dispositivos para a troca de mensagens, não assumindo uma conetividade fim a fim entre um emissor e um recetor mas sim com base em oportunidades de contacto. Assim, a comunicação é *multihop*, uma vez que cada nó intermediário armazena a mensagem até surgir uma oportunidade de contacto e aí fazer o encaminhamento da mensagem.

Um dos desafios que se levantam neste contexto é avaliar qual a real oportunidade para a troca de mensagens entre os dispositivos, onde se inclui a avaliação da conectividade entre dispositivos afastados, a largura de banda disponível, e os atrasos na comunicação. Obviamente, o conhecimento da rede (a probabilidade de encontros entre nós, a frequência com que cada lugar é visitado, etc.) pode ajudar a prever futuras oportunidades de contacto e identificar melhores candidatos para retransmitir os dados para o destino.

Atualmente já é possível avaliar o desempenho destas redes através de modelos analíticos e de simulação. No entanto, o desempenho destas redes em ambientes reais ainda não é completamente conhecido.

1.2 Objetivos

Com este trabalho pretende-se explorar a disponibilidade de dados reais sobre o movimento de pessoas para avaliar o desempenho de redes oportunistas. A análise de *traces* Wi-Fi pode fornecer informações importantes sobre o assunto, em particular, para estimar o atraso e a largura de banda. Entre outros, um dos grandes desafios a abordar é a grande

quantidade de dados a analisar (*big data*), onde serão estudadas técnicas para analisar grandes volumes de dados de forma eficiente.

Face a isto, neste estudo foram definidos os seguintes objetivos:

- Avaliar o desempenho das redes oportunistas em ambientes reais;
- Desenhar algoritmos que sejam adequados à avaliação do desempenho destas redes com base em *traces*;
- Explorar as capacidades de processamento dos sistemas de gestão de bases de dados, combinadas com linguagens de programação orientadas a objetos para o desenvolvimento de ferramentas eficientes de processamento de dados;
- Validar os algoritmos desenvolvidos anteriormente.

1.3 Estrutura da dissertação

A dissertação encontra-se organizada em oito capítulos. No presente capítulo, correspondente ao primeiro, é feito um enquadramento ao estudo realizado, apresentando os principais objetivos propostos e uma descrição geral da organização da escrita do documento. No segundo capítulo apresenta-se uma breve revisão bibliográfica, no qual se destacam três processos utilizados na avaliação de redes oportunistas, modelos de mobilidade, ferramentas de simulação e *traces* de dados reais de mobilidade. Por fim, neste capítulo é igualmente apresentado um resumo dos principais protocolos de encaminhamento utilizados e propostos na área das redes oportunistas. No capítulo 3 é apresentada a abordagem metodológica adotada e planeada de forma a cumprir todos os objetivos do trabalho. No capítulo 4 é apresentado detalhadamente todo o desenvolvimento de cada algoritmo, que inclui a sua descrição genérica e em pseudocódigo, a sua implementação, os testes de correção e desempenho e os respetivos resultados alcançados nos testes. No capítulo 5 são apresentadas todas as experiências realizadas com o algoritmo de encaminhamento epidémico utilizando os dados disponíveis. No capítulo 6 são apresentados os resultados obtidos e toda a discussão associada aos mesmos. No capítulo 7 são resumidas as principais considerações alcançadas neste estudo e as perspetivas de trabalho futuro. Finalmente é apresentada a listagem de toda a bibliografia utilizada na execução deste trabalho e os anexos inerentes a todo o trabalho.

Ao longo da dissertação são utilizados indiferentemente os termos ponto de acesso e AP, tendo ambos o mesmo significado. Sob o mesmo ponto de vista, são utilizados igualmente os termos nó móvel e station referindo-se ao mesmo conceito. Os termos *logs* e registos são também utilizados indiferentemente em algumas ocasiões.

CAPÍTULO 2
REVISÃO BIBLIOGRÁFICA

Ao longo dos últimos anos, a comunidade de investigação tem-se debruçado cada vez mais na área das redes oportunistas, levando a um considerável aumento de interesse no seu estudo. Esta área tem sido considerada como uma evolução natural das redes *Ad Hoc* móveis, normalmente conhecidas como MANETs.

Inicialmente, os estudos centraram-se no funcionamento base destas redes, tentando perceber essencialmente o modo como operam. Como foi dito anteriormente, neste tipo de redes os nós são capazes de comunicar uns com os outros sem que exista um meio físico que os una diretamente. Um exemplo prático que pode ser aplicado ao funcionamento destas redes é quando viajamos num autocarro. Durante essa mesma viagem estamos fisicamente próximos de outros passageiros e o nosso telefone móvel pode comunicar com os dos restantes passageiros para trocar informação. Este exemplo pode caracterizar o funcionamento base de uma rede oportunista.

Estes estudos iniciais foram relevantes pois foi possível identificar casos de uso e cenários de avaliação adequados. Após isso, as últimas investigações têm sido dirigidas ao estudo do desempenho destes sistemas, não só através de modelos analíticos e de simulação, bem como através de *traces* de dados reais. Como esta área ainda está em pleno crescimento, as investigações sobre o desempenho de uma rede oportunista são essenciais para perceber realmente a importância que estas redes podem ter no presente e futuro das comunicações. A conetividade intermitente entre os nós móveis torna estas investigações num trabalho cada vez mais desafiante.

Na última década, para além destes estudos, os investigadores também têm-se focado muito no desenvolvimento de protocolos e modelos de mobilidade eficientes. O desenvolvimento de novos e melhores métodos de avaliação de desempenho destes sistemas, de novos modelos teóricos e matemáticos são outras das frequentes pesquisas atuais.

Assim, atualmente a avaliação do desempenho de uma rede oportunista e dos respetivos protocolos é feita essencialmente através de três abordagens: modelos de mobilidade, ferramentas de simulação e *traces* de dados reais.

Ao longo desta secção apresenta-se uma visão global do estado da arte na avaliação do desempenho de redes oportunistas. São descritas cada uma das abordagens referidas e são identificadas diversas publicações já existentes na literatura sobre o assunto. Por fim, é feito um

levantamento dos principais protocolos existentes nesta área, dando igualmente destaque a várias publicações onde são propostos novos protocolos e apresentadas análises e comparações.

2.1 Modelos de Mobilidade

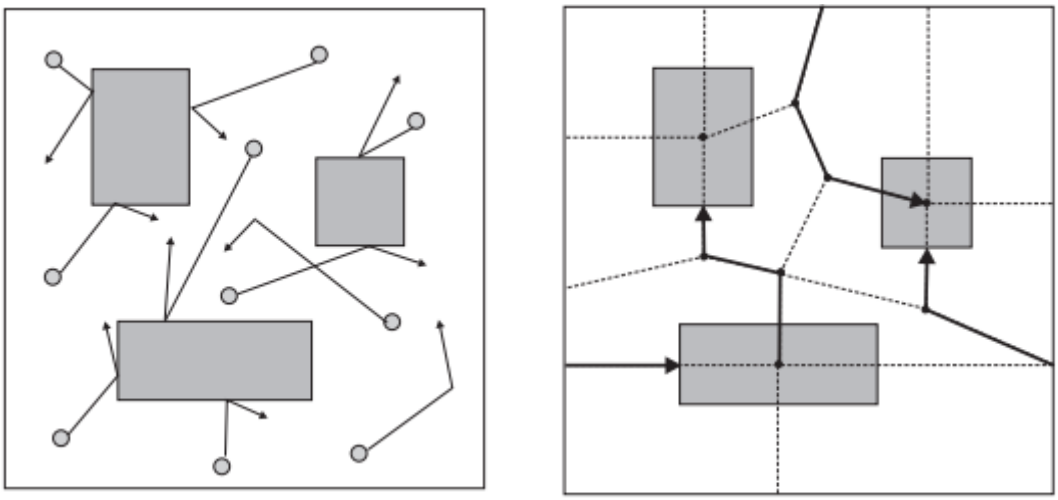
Um modelo de mobilidade pretende descrever os movimentos dos utilizadores móveis, e a forma como a sua localização, velocidade e aceleração mudam ao longo do tempo, através da definição de determinados parâmetros que caracterizam o ambiente. Apesar de por vezes ser difícil, é desejável que os modelos se aproximem o mais possível do padrão de movimento do mundo real. Se tal não acontecer, os resultados alcançados em diversos estudos podem levar a que sejam retiradas conclusões erradas.

Em diversos estudos, os investigadores preferem os modelos de mobilidade sintéticos, uma vez que estes possibilitam configurar vários parâmetros da simulação e reproduzir os resultados facilmente (Thabotharan Kathiravelu e Arnold Pears, 2011).

A abordagem a partir de modelos de mobilidade tem ganho cada vez mais importância e interesse nos últimos anos. Alguns estudos iniciais com estes modelos revelaram que a escolha do modelo de mobilidade pode ter um efeito significativo na avaliação do desempenho de um protocolo de rede (Camp et al., 2002). Neste estudo é feita uma avaliação de sete diferentes modelos de mobilidade, excluindo os modelos mais recentes, dando especial atenção aos dois modelos mais comuns e mais utilizados pelos investigadores, o *Random Walk* (RW) e o *Random Waypoint* (RWP). Os resultados alcançados no estudo permitem perceber o efeito que a escolha de um modelo de mobilidade tem no desempenho de um protocolo. Ao ser utilizado o mesmo modelo mas com diferentes parâmetros configurados também pode fazer variar o desempenho de um determinado protocolo. Outra conclusão apresentada pelos autores é o facto de que a escolha do modelo de mobilidade deverá ser feita de acordo com o cenário do mundo real que se espera representar.

Atualmente, a comunidade de investigação procura desenvolver e avaliar novos modelos de mobilidade que consigam representar, o melhor possível, ambientes do mundo real e respetivos padrões de movimento. Um desses exemplos é a proposta de criação de modelos ainda mais realistas, através da integração de obstáculos no caminho dos nós, restringindo o movimento dos mesmos e a própria transmissão sem fios (Jardosh et al., 2003). Os resultados obtidos neste

estudo mostram que a utilização de obstáculos tem um impacto significativo no desempenho de um protocolo de rede. A figura 2.1 mostra dois exemplos do movimento com esses obstáculos.



(a) Movimento com obstáculos

(b) Movimento com obstáculos e caminhos pré-definidos

Figura 2.1 – Exemplo de movimento baseado em obstáculos (Jardosh et tal., 2003).

Outro dos modelos propostos é o *Weighted Waypoint* (Hsu et al., 2005). Neste modelo é feita a recolha das preferências na escolha de destinos para caracterizar padrões de mobilidade de pedestres num ambiente de campus, ou seja, a escolha do destino não é feita aleatoriamente pelos pedestres mas sim com base em locais de interesse. A modelação deste comportamento é feita através da definição de locais de interesse e da atribuição de valores a cada um desses locais, representando o peso da probabilidade do local ser escolhido como destino. Neste modelo, a probabilidade de escolha de um destino é determinada pela localização atual e pelo tempo.

O modelo MobiREAL baseia-se em regras probabilísticas para descrever as mudanças no comportamento dos nós móveis (destinos, rotas e velocidades/direções), considerando que essas mudanças devem-se ao ambiente à sua volta (Konish et al., 2005). Os autores desenvolveram um simulador baseado neste modelo, tendo disponibilizado vários cenários de simulação na página web do projeto.

Um método para extrair um modelo de mobilidade através de *traces* de utilizadores reais é igualmente proposto em (Kim et al., 2006). Utilizando este método, foram extraídas faixas de um *trace* de 13 meses e analisadas as informações extraídas, como a velocidade ou os tempos de pausa. Estas informações permitiram formar um modelo analítico, que foi validado através de uma comparação de desempenho, entre as localizações presentes nos *traces* e as localizações que foram determinadas por utilizadores com dispositivos GPS e 802.11.

À medida que novos modelos são propostos, surgem novos pontos de vista e novas técnicas. Uma nova técnica tem sido aplicada no desenvolvimento de modelos de mobilidade. Esta técnica é baseada na teoria das redes sociais, ou seja, nas relações sociais entre as pessoas (Musolesi et al., 2004; Musolesi e Mascolo, 2006 e 2007).

Outro estudo relevante também aborda as relações sociais como caraterística do movimento das pessoas (Boldrini et al., 2011). Neste estudo, são comparados modelos de mobilidade relativamente a três propriedades do movimento humano: espacial, temporal e social. Através do mesmo, pode-se constatar abordagens a novos modelos de mobilidade que exploram a parte social das pessoas no contexto das redes oportunistas (Musolesi e Mascolo, 2007; Borrel et al., 2009; Boldrini e Passarella, 2010; Yang et al.,2010; Fischer et al., 2010).

2.2 Ferramentas de simulação

Uma das formas mais comuns de avaliar o desempenho de redes e os seus respetivos protocolos é através de simulações. Atualmente existem inúmeros simuladores de redes disponíveis, como por exemplo, o ns-3 (Henderson et al., 2006) ou o OMNet++ (Varga, 2001; Varga e Hornig, 2008), cada um com as suas próprias vantagens e desvantagens.

Especificamente para redes oportunistas, existe um simulador que facilita as simulações nesta área, sendo um dos simuladores mais conhecidos e utilizados em várias estudos académicos e científicos. Esse simulador é o The ONE e foi desenvolvido em projetos suportados pela *Nokia Research Center* (Keränen et al., 2009).

Alguns autores utilizam um determinado simulador mas alteram-no de forma a adapta-lo aos seus trabalhos. É o caso do simulador OMNet++, já citado anteriormente (Helgason e Jonsson, 2008). Neste estudo, os investigadores alteraram o simulador na realização do trabalho, de forma a simular os contactos em redes oportunistas. Para tal, seguiram essencialmente dois métodos. No primeiro que denominaram de *mobility-driven method*, um *trace* de mobilidade especificava os padrões de mobilidade dos nós. Já no segundo método, denominado *contact-driven method*, foi utilizado um *trace* de contactos que continha os tempos de contacto dos nós e as respetivas durações.

Para além destes investigadores, outros utilizaram igualmente o simulador OMNEt++ nas suas pesquisas, mais especificamente para avaliar o desempenho do protocolo que apresentavam (Musolesi e Mascolo, 2008). O protocolo apresentado por estes autores, denominado de CAR (*Context-Aware Routing*), utiliza o modelo de mobilidade apresentado anteriormente pelos mesmos (Musolesi et al., 2004; Musolesi e Mascolo, 2006). A figura 2.2 mostra um exemplo de um cenário de simulação no OMNEt++.

Figura 2.2 – Exemplo de cenário de simulação no OMNEt++ (Helgason e Jonsson, 2008).

Como foi dito anteriormente, uma rede oportunista é normalmente considerada uma extensão de uma MANET. Assim, à medida que crescia o interesse dos investigadores nesta extensão, os simuladores e os modelos de mobilidade utilizados nas MANETs foram sendo adotados nas investigações das redes oportunistas (Kathiravelu e Pears, 2006). Neste estudo, os autores utilizaram o simulador JiST/SWANS, *Java in Simulation Time/ Scalable Wireless Ad hoc Network Simulator* (Barr et al., 2005). Este simulador, que é construído por cima do simulador JiST, permite aos investigadores simularem diferentes cenários de contactos sem fios com diferentes modelos de mobilidade. A figura 2.3 mostra a arquitetura do sistema JiST.

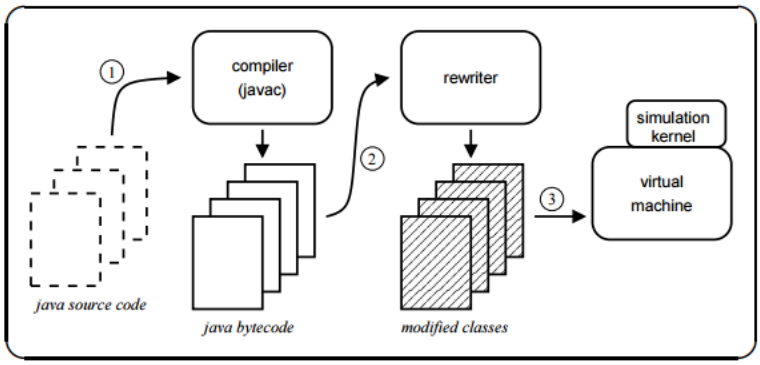


Figura 2.3 – A arquitetura do sistema JiST (Barr et al., 2005).

À medida que vários simuladores iam sendo utilizados em várias pesquisas, alguns investigadores optaram por desenvolver os seus próprios simuladores (Leguay et al., 2006; Heinemann et al., 2008).

Embora as simulações sejam uma ferramenta comum e útil nas pesquisas, podem apresentar alguns erros (Kurkowski et al., 2004 e 2005). Nestes estudos, os autores referem que apesar de haver um aumento de simulações nas pesquisas, há também uma diminuição na credibilidade dos resultados das simulações. No estudo de 2004 foi avaliada a credibilidade de vários simuladores existentes até aquele momento. Para tal, os autores focaram-se em 4 áreas de credibilidade numa pesquisa: a repetição, a imparcialidade, o realismo e a estatística.

Outros investigadores, também já identificaram alguns equívocos através de estudos de simulação (Andel e Yasinsac, 2006; Grasic e Lindgren, 2012). Em particular, estes últimos autores já publicaram inúmeros estudos onde apresentam alguns desses equívocos.

2.3 *Traces*

A utilização de *traces* também tem ganho cada vez mais atenção por parte da comunidade de investigação na área das MANETs e das redes oportunistas. Estes *traces* têm o objetivo de caracterizar os padrões de contacto entre os dispositivos, para a validação de novos protocolos e aplicações. Normalmente, a recolha dos mesmos é feita recorrendo aos participantes de cenários específicos, tais como conferências, *campus* académicos, entre outros.

Muitos investigadores têm realizado estudos com *traces*, pois têm a convicção que os mesmos podem revelar factos essenciais acerca dos contactos entre dispositivos (Chaintreau et al., 2005; LeBrun et al., 2006; Leguay et al., 2006). A figura 2.4 mostra uma equipa de investigação a preparar uma recolha de dados de contactos oportunistas numa conferência

académica em Dezembro de 2007. Esta recolha foi feita utilizando pequenos dispositivos móveis com conectividade sem fios Bluetooth.

Figura 2.4 – Equipa de investigação a preparar uma recolha de dados (Thabotharan Kathiravelu e Arnold Pears, 2011).

O verdadeiro início do estudo da possibilidade de encaminhar dados em oportunidades de contacto através de *traces* de dados reais foi elaborado por uma equipa de investigadores específica (Su et al., 2004). Neste estudo os autores registaram os dados utilizando PDAs. Apesar deste estudo, o primeiro trabalho concreto sobre esta forma de registar traces, incluindo a análise das suas propriedades e da possibilidade de encaminhamento através de contactos oportunistas foi desenvolvido por outros investigadores no ano seguinte (Chaintreau et al., 2006). Estes autores analisaram três *traces* disponíveis ao público, mais especificamente *traces* da Universidade de Toronto, no Canadá, recolhidos pelos autores do estudo inicial (Su et al., 2004), da Universidade da Califórnia (San Diego) (McNett e Voelker, 2005) e da Universidade de Dartmouth (Henderson et al., 2004), ambas dos EUA. Cada um destes *traces* diferem em algumas características, especialmente em termos da duração temporal da recolha dos dados. O primeiro *trace* referente à Universidade de Toronto foi recolhido durante 16 dias através de 20 PDAs habilitados da tecnologia *Bluetooth* e transportados por estudantes. A Universidade da Califórnia recolheu o seu *trace* através da rede Wi-Fi, onde foram incluídos registos de ligação a Pontos de Acesso (APs) durante 3 meses. Por fim, o *trace* da Universidade de Dartmouth, igualmente recolhido da rede Wi-Fi, inclui registos SNMP a partir de APs por um período de 4 meses.

Para além da utilização dos referidos traces, a mesma equipa de investigação decidiu recolher os seus próprios dados, recorrendo a dispositivos iMote, ilustrado na figura 2.5 (Chaintreau et al., 2005). Basicamente, um dispositivo iMote é uma plataforma desenvolvida pela Intel para recolher *traces* através da tecnologia sem fios Bluetooth (Chaintreau et al., 2005; Leguay

et al., 2006). Como foi referido anteriormente, PDAs e até mesmo *smartphones* são outras das possibilidades para este tipo de atividade. Apesar de terem um custo elevado em grande escala são boas opções em termos de bateria e memória (Keshav et al., 2007).

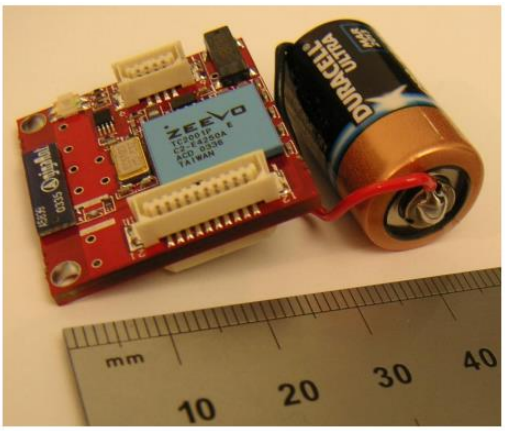


Figura 2.5 – Dispositivo iMote conectado a uma bateria CR2 (Chaintreau et al., 2005).

Através dos dispositivos iMote os investigadores recolheram 3 conjuntos de traces. Dois desses *traces* (Cambridge e Intel) foram recolhidos em laboratório, onde investigadores e alunos de doutoramento transportavam os referidos dispositivos. O terceiro foi durante a conferência IEEE INFOCOM de 2005, quando os dispositivos eram transportados pelos participantes da conferência.

A partir do momento em que Chaintreau *et al.* publicaram o seu trabalho sobre a recolha e análise de *traces*, muitos investigadores seguiram-nos. Um desses exemplos é o estudo feito na cidade de Cambridge com os mesmos dispositivos iMote (Leguay et al., 2006). Estes dispositivos foram transportados por estudantes ou fixados em determinados locais da cidade e recolheram os dados dos contactos durante 13 dias. O objetivo principal deste estudo era verificar se era viável uma rede de distribuição de conteúdo em toda a cidade.

Outros autores optaram por alargar o horizonte desta abordagem ao investigarem a viabilidade da disseminação de dados utilizando autocarros em trânsito na Universidade da Califórnia (LeBrun e Chuah, 2006). Cada autocarro estava equipado com uma cache de distribuição de conteúdos, dotada de uma interface na tecnologia Bluetooth, denominada de BlueSpot. Assim, cada dispositivo que possuísse uma interface Bluetooth podia conectar-se a essa cache e descarregar informação, enquanto o autocarro estava em movimento.

Atualmente, a base de dados CRAWDAD é um bom e conhecido repositório de *traces* WiFi (Yeo et al., 2006). Vários investigadores utilizam estes *traces* nos seus estudos para extraírem informação e desenvolverem novos modelos de mobilidade e novos protocolos. Na tabela 2.1 são

apresentados alguns exemplos desses *traces*, estando os mesmos disponíveis publicamente na referida base de dados.

Outros investigadores utilizaram igualmente *traces* desta fonte nos seus estudos. De entre muitos exemplos, podem destacar-se dois mais relevantes. O primeiro estudo a destacar foca-se num conceito mais particular e importante acerca da interação entre nós móveis: o conceito de encontro (Hsu e Helmy, 2010). Segundo estes investigadores, um encontro é definido como o intervalo de tempo em que dois nós móveis estão associados ao mesmo AP. Este tipo de evento é denominado de encontro indireto. Esta definição não é clara nem exata, pois muitas vezes os nós estão próximos uns dos outros mas podem estar associados a outros APs, ou podem estar fora da área de cobertura dos APs, ou até mesmo estarem envolvidos em eventos de ping-pong (Kim et al., 2006; Jahromi et al., 2014). Um encontro do tipo direto ocorre quando dois dispositivos móveis estão no alcance de comunicação um do outro (Jahromi et al., 2014). Outro estudo a destacar refere-se a uma avaliação e comparação, não só de 5 protocolos existentes, mas também do protocolo que é proposto pelos autores no referido estudo (Song e Kotz, 2007). Por fim, destaca-se também um estudo onde os autores utilizam igualmente vários *traces* do repositório CRAWDAD para avaliar uma estratégia de encaminhamento que propõem (Radenkovic e Grundy, 2011).

Tabela 2.1 – Propriedades de alguns *traces* de conectividade oportunista acessíveis ao público [adaptado de Thabotharan Kathiravelu e Arnold Pears, 2011]. Cambridge1 [Leguay et al., 2006]; Intel, Cambridge2 e Infocom [Chaintreau et al., 2005].

Propriedades do dispositivo	**Cambridge1**	**Intel**	**Cambridge2**	**Infocom**
Dispositivo	iMote	iMote	iMote	iMote
Tecnologia de Acesso	Bluetooth	Bluetooth	Bluetooth	Bluetooth
Duração (dias)	13	3	5	3
Granularidade (segundos)	120	120	120	120
Nº dispositivos internos	54	9	12	41
Nº dispositivos Externos	11357	118	203	233
Nº Contactos	8545	2766	6732	28216

A análise deste tipo de *traces* tinha como objetivo perceber as limitações da transmissão de dados através das oportunidades de contacto. Assim, foram definidos dois novos termos: *Contact Time* e *Inter-Contact Time*. O primeiro refere-se ao intervalo de tempo em que dois dispositivos móveis estão no alcance de comunicação um do outro. O segundo ao intervalo de tempo entre dois contactos consecutivos, de dois dispositivos móveis que estão em contacto um com o outro de forma intermitente (Thabotharan Kathiravelu e Arnold Pears, 2011). Os dois contactos consecutivos referem-se ao mesmo par de dispositivos. Na figura 2.6, é possível visualizar uma representação destes dois termos.

Estes termos têm sido identificados em várias investigações como sendo as métricas mais importantes no estudo de redes oportunistas (Chaintreau et al., 2005; Zheng et al., 2009; Zyba et al., 2011). Na primeira investigação referida, os autores mostram que 90% dos contactos duram menos que 10 minutos, o que significa que os nós envolvidos nos contactos têm pouco tempo para trocarem dados uns com os outros. Esta situação dá ênfase à necessidade de serem utilizados protocolos de contacto eficientes.

Outra perspetiva baseada em *traces* é proposta por outros autores. Eles propõem uma ferramenta para gerar automaticamente *traces* de conectividade (CTG) (Calegari et al., 2007). A ferramenta tem como entrada *traces* de mobilidade real e é capaz de produzir um conjunto de *traces* com propriedades de conectividade semelhantes mas com parâmetros escalados, como por exemplo o número de nós. Com isto, os investigadores podem melhorar a análise do desempenho de protocolos e aplicações.

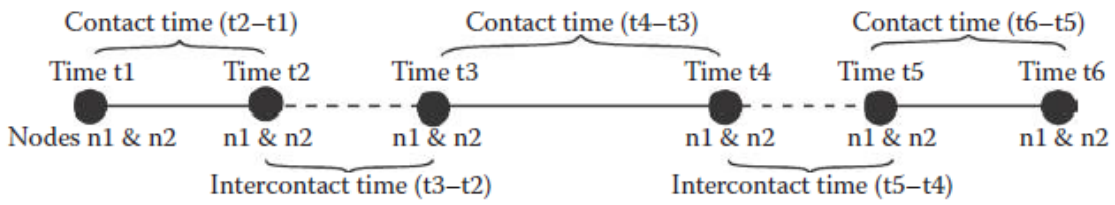


Figura 2.6 – Caracterização do *Contact-Time* e *Inter-Contact Time* para dois nós n1 e n2 em contacto intermitente (Thabotharan Kathiravelu e Arnold Pears, 2011).

Um estudo particular fornece uma visão geral de modelos de mobilidade sintéticos e baseados em t*races*, sublinhando a necessidade de surgirem *traces* e modelos de mobilidade mais realistas (Aschenbruck et al., 2011).

É um facto que esta utilização e recolha de *traces* tem auxiliado os investigadores no desenvolvimento de algoritmos e estratégias de encaminhamento adequadas para redes

oportunistas (Chaintreau et al., 2007). Contudo é essencial referir que mesmo utilizando *traces* de dados reais, os mesmos podem levar a resultados sobrestimados. Foi desenvolvida uma aplicação de teste para redes oportunistas que mostrou que a taxa de entrega e o atraso são superestimados em simulações quando estes seguem pressupostos irrealistas (Ristanovic et al., 2012).

Também durante a recolha dos dados podem surgir alguns problemas, nomeadamente com os dispositivos. Um estudo aponta algumas dessas dificuldades, como a possibilidade dos dispositivos ficarem sem bateria e apresentarem problemas em termos de memória durante o processo de recolha dos dados (Leguay et al., 2006).

Para além disso, em termos mais práticos, ao recolher-se dados através de um *scanning* de *Bluetooth* durante uma experiência num determinado cenário, são retornados todos os dispositivos encontrados munidos desta tecnologia, contudo há que contar com um elevado número de dispositivos que são descobertos mas que não participam na experiência (Vinicius et al., 2014). Assim, como forma de beneficiar com tal situação, estes *traces* e as suas propriedades levaram a que fossem largamente utilizados no estudo e avaliação de novos protolocos de encaminhamento oportunistas, com base sobretudo nas propriedades das redes sociais, já referidas anteriormente. Como resultado do estudo desta interação Social (nós que permanecem por muito tempo no mesmo lugar) foi demonstrado que a densidade é mais importante do que o próprio comportamento social, contando também para o estudo as pessoas que vagueiam sem um rumo determinado (Zheng, et al., 2009).

2.4 Protocolos

Um protocolo de encaminhamento é desenhado para encaminhar mensagens de um nó (origem) para um outro nó (destino). Ao contrário das redes clássicas, as redes oportunistas são caracterizadas pelas frequentes quebras nas ligações e pelos elevados atrasados na comunicação. Face a isto, os paradigmas clássicos relativos ao encaminhamento de mensagens não podem ser aplicados a estas redes, pois as redes clássicas são baseadas no estabelecimento de um caminho fim a fim entre a origem e o destino. Como consequência disso, o encaminhamento de mensagens nas redes oportunistas (OppNets) tem que ser suportado por protocolos específicos, que sejam eficientes nos ambientes proporcionados pelas mesmas.

Como refere Song e Kotz no seu estudo (Song e Kotz, 2007), nas redes móveis oportunistas os dispositivos móveis podem ser transportados por pessoas (Burgess et al., 2006),

veículos (Campbell et al., 2006) ou até mesmo por animais (Juang et al., 2002). Alguns dispositivos, quando se movem perto uns dos outros, podem formar uma pequena rede móvel *ad hoc* mas, como referem os autores, à semelhança dos protocolos clássicos mencionados anteriormente, também os protocolos de encaminhamento *ad hoc* assumem a existência de um caminho fim a fim, falhando assim nas redes oportunistas. Exemplos desses protocolos de encaminhamento *ad hoc* são o AODV (Perkins e Royer, 1999) ou o DSDV (Perkins e Bhagwat, 1994).

Qualquer nó presente na rede pode não só gerar mensagens para qualquer um dos outros nós como também servir de nó intermediário, mantendo as mensagens que se destinam a outros nós para posteriormente serem encaminhadas. O encaminhamento de mensagens nas OppNets é feito durante o contacto entre dois nós, conceito esse já abordado anteriormente e que foi caracterizado por ser o período de tempo em que dois nós têm a oportunidade para comunicarem. É importante referir que um nó pode estar em contacto com vários nós diferentes ao mesmo tempo.

Existem duas abordagens principais que podem ser consideradas como soluções para a entrega de mensagens neste tipo de redes. A primeira passa pelo nó de origem esperar passivamente para estar no raio de comunicação do nó de destino e aí entregar a mensagem. A outra passa pelo nó de origem inundar todos os nós que estão no seu raio de comunicação com a mensagem a transmitir. Os nós que recebem essa mensagem inundam posteriormente a rede da mesma forma, repetidamente. Como referem novamente Song e Kotz no seu estudo, estas duas abordagens possuem vantagens e desvantagens óbvias: a primeira solução pode apresentar uma baixa taxa de entrega mas com utilização de poucos recursos; a segunda, contrariamente à primeira, pode apresentar uma alta taxa de entrega, utilizando muitos recursos.

Ao longo dos anos, outros protocolos de encaminhamento oportunista tem sido propostos e avaliados por vários investigadores. Apesar de haver um crescimento nos últimos anos, poucos protocolos foram avaliados em ambientes realistas, apesar de serem utilizados alguns modelos de mobilidade aleatórios mais conhecidos, como o *Random Walk* ou o *Random Waypoint*. Após os investigadores perceberem algumas limitações destes modelos de mobilidade, alguns deles começaram a estudar os protocolos de encaminhamento através de *traces* de dados reais. Por exemplo, foi analisado o impacto de alguns algoritmos de encaminhamento ao longo de dois tipos de *data sets*, um composto por *traces* disponíveis ao público que refletiam a conectividade entre clientes e APs em vários redes Wi-Fi; e o outro composto por *traces* recolhidos pelos próprios

autores através de dispositivos iMotes (Chaintreau et al., 2007). Outro exemplo de estudo foi a simulação de um conjunto de protocolos de encaminhamento numa pequena rede experimental (Su et al., 2006). Estes estudos permitiram aos investigadores entenderem não só os limites teóricos das próprias redes oportunistas mas também o desempenho de um protocolo de encaminhamento de uma pequena rede com cerca de 20 a 30 nós (Song e Kotz, 2007).

Para além destes dois estudos, muitos outros foram apresentados. De forma a uma melhor organização e compreensão, optou-se por dividir os principais protocolos de encaminhamento em redes oportunistas em 3 grupos, à semelhança do que apresenta Mota et al. no seu estudo (Mota et al., 2014). "Nenhum Contexto" engloba os protocolos que encaminham uma mensagem na rede, sem possuírem previamente qualquer conhecimento da mesma. "Contexto Parcial" engloba os protocolos que utilizam apenas informações da rede no encaminhamento de mensagens, como por exemplo, o estado dos vizinhos e o histórico de contactos. Por fim, "Contexto Completo" inclui os protocolos que utilizam o estado dos vizinhos e dos atributos de rede para tomar as melhores decisões de encaminhamento. Na tabela 2.2, adaptada dos mesmos autores, estão classificados os principais protolocos abordados seguidamente.

2.4.1 Nenhum Contexto

Os protocolos que são apresentados de seguida não possuem qualquer conhecimento da rede, como a topologia da mesma ou a frequência de encontros entre os nós. Os nós ao encaminharem a mensagem baseiam-se meramente no seu próprio estado.

2.4.1.1 Epidémico

O protocolo epidémico baseia-se no processo de inundação de mensagens na rede (Vahdat e Becker, 2000). O nó de origem envia uma cópia da mensagem a todos os nós com que se encontre. Posteriormente, esses nós que recebem a mensagem procedem da mesma forma, enviando uma cópia da mensagem recebida a todos os nós com que se encontrem. Desta forma, possivelmente, uma cópia da mensagem chegará ao destino da mensagem.

Segundo os autores, o objetivo deste protocolo é maximizar a entrega das mensagens e minimizar o atraso. Através de uma implementação no simulador Monarch, os autores mostram que a utilização deste protocolo pode eventualmente atingir uma taxa de entrega de 100% das mensagens. Aliada a esta entrega, os recursos consumidos foram razoáveis num conjunto de cenários selecionados pelos autores.

Apesar disso e apesar deste protocolo ser simples, a questão dos recursos utilizados pelo mesmo na entrega das mensagens é muito debatida. Song e Kotz dão vários exemplos dos problemas enfrentados na utilização deste protocolo (Song e Kotz, 2007). Inicialmente, os autores referem que a comunicação em excesso por parte dos vários nós pode rapidamente descarregar a bateria de cada um deles, o que leva este protocolo a ter uma baixa taxa de entrega de mensagens quando são utilizados dispositivos limitados em temos de recursos. Para além disso, referem que como cada nó mantém uma cópia de cada mensagem, o armazenamento não é usado eficientemente e a capacidade da rede é limitada.

De forma a tentar contornar tais situações, os autores mencionam que cada nó deve expirar mensagens depois de algum período de tempo ou parar de encaminhá-las depois de um certo número de saltos. Após uma mensagem expirar, a mensagem não será enviada e será eliminada do armazenamento de qualquer nó que a contenha. Por fim, de forma a reduzir o custo da comunicação entre os nós, é referida a transferência de mensagens de índice antes de enviar qualquer mensagem de dados. Estas mensagens de índice contêm Ids de cada mensagem que um nó tem atualmente e verificando estas mensagens, um nó só transfere mensagens que ainda não estão armazenadas nos outros nós.

Este protocolo é regularmente utilizado pela comunidade científica como ferramenta de avaliação de propostas de protocolos de encaminhamento, avaliando por exemplo a sua eficiência na entrega de mensagens e o atraso.

Tabela 2.2 – Classificação de protocolos de encaminhamento em redes oportunistas [adaptada de Mota et al., 2014].

Ano	Classe	Protocolo	Principal Caraterística	Desvantagens
2000	Nenhum Contexto	Epidémico	Maximiza a taxa de entrega de mensagens e minimiza a latência.	Alta sobrecarga.
2004		Cópia única	Baixa sobrecarga.	Alto atraso e baixas probabilidades de entrega.
2005		*Spray and wait*	Espalha apenas L cópias de uma mensagem.	Aumenta o atraso.
2003	Contexto Parcial	*Prophet*	Baseado no histórico de encontros.	Trabalha similarmente ao Epidémico em cenários densos.
2005		CAR	Trocas de probabilidade de entrega multi-objetivo.	Precisa de tempo para aprender o contexto.
2010		HYMAD	Protocolo hibrido de *ad hoc* e encaminhamento oportunista.	Testado apenas em cenários muito densos.
2007		*HiBOp*	Trocas de interesses dos utilizadores.	Precisa de tempo para aprender o contexto.
2007	Contexto Completo	*SimBet*	Centralidade egocêntrica e as suas semelhanças sociais.	Pode sobrecarregar os nós com alta centralidade.
2008		*BubbleRap*	Grupos e classificação dos nós dentro de uma comunidade.	Pode não funcionar em redes aleatórias.

2.4.1.2 *Spray and wait*

O protocolo *Spray and wait* baseia-se em duas fases principais: na primeira denominada de *Spray*, são espalhadas L cópias de mensagens geradas pelo nó de origem para L nós distintos; na segunda fase, denominada de *Wait*, se o destino da mensagem não for encontrado inicialmente na primeira fase, cada nó que recebeu a mensagem armazena-a e apenas a transmite diretamente para o destino da mesma (Spyropoulos et al., 2005). Os autores propõem um conjunto de variações do protocolo, diferindo-os no modo como as L cópias iniciais da mensagem são espalhadas na rede. A mais eficiente das variações denomina-se *Binary Spray and wait* e consiste num determinado nó com um dado número de mensagens, transmitir para cada nó com que se encontre metade das mensagens distintas que possui.

Para além destes dois algoritmos, os algoritmos de cópia única também poderiam ser inseridos neste grupo de protocolos de nenhum contexto (Spyropoulous et al., 2004).

2.4.2 Contexto Parcial

Os próximos protocolos apresentados recolhem informação sobre os seus encontros com outros nós, utilizando essa informação para selecionar o melhor nó para encaminhar uma mensagem para o destino.

2.4.2.1 *Prophet*

O *Prophet* é o protocolo de encaminhamento probabilístico que utiliza o histórico de encontros anteriores e a transitividade, que é usada para estimar a probabilidade de entrega de cada nó para outro nó (Lindgren et al., 2003). Os autores utilizam uma métrica denominada previsibilidade de entrega que indica a quão provável é que um nó será capaz de entregar uma mensagem a um determinado destino. Isto significa que o movimento do nó não é aleatório mas sim baseado em padrões de mobilidade. Se um nó visita frequentemente uma determinada parte da rede, assume-se que existe uma elevada probabilidade do nó visitar esse mesmo lugar no futuro.

Diversas simulações foram feitas através de um simulador personalizado pelos autores, de forma a estabelecer uma comparação entre este protocolo e o protocolo epidémico. Como resultado dessa comparação, os autores mostram que o *Prophet* é capaz de entregar mais mensagens do que o epidémico com uma sobrecarga de comunicação menor. À semelhança do

protocolo epidémico, também o *Prophet* é largamente utilizado como base para a avaliação de novas propostas de protocolos.

2.4.2.2 CAR

O protocolo CAR tem como filosofia de processo prever se um dado destino pertence à mesma parte da rede do nó remetente da mensagem ou do nó de retransmissão (Mascolo e Musolesi, 2006). Os autores utilizam duas informações específicas como contexto, nomeadamente o número de vizinhos de um nó e o seu atual nível de energia. Para executarem as simulações, utilizaram o simulador OMNeT++ e definiram como métricas a taxa de entrega de mensagens, a sobrecarga na troca de contexto e o atraso.

Segundo os autores desta classificação de protocolos, Mota et al., este protocolo foi considerado de contexto parcial, uma vez que para reunir as informações de rede de um nó não são exigidas trocas de mensagens explícitas. Se o destino está na parte da rede do nó que transmite a mensagem, a mensagem é transmitida utilizando o DSDV (encaminhamento ad hoc). Se não, é utilizada uma função baseada na taxa de variação da conetividade e na probabilidade de o destino agregado na mesma parte da rede do nó de retransmissão.

2.4.2.3 HYMAD

O protocolo HYMAD é definido pelos autores como sendo um protocolo de encaminhamento DTN-MANET híbrido que usa o paradigma DTN na comunicação entre grupos disjuntos de nós, enquanto está a usar encaminhamento MANET dentro desses grupos (Whitbeck e Conan, 2009). O encaminhamento MANET tende a ter uma menor sobrecarga em grupos de nós altamente ligados. Os autores avaliaram o protocolo através da utilização de um *trace* com elevada densidade e conectividade entre os nós. No cenário utilizado, o HYMAD apresenta um bom desempenho em comparação com o Epidémico. Apesar de tais resultados, há que realçar que os autores não avaliaram o desempenho deste protocolo noutros cenários, nomeadamente num que apresentasse uma densidade de nós reduzida.

2.4.3 Contexto completo

Os protocolos de contexto completo seguidamente apresentados utilizam diversas informações para encaminhar as mensagens, tais como o ambiente em volta, o histórico de contactos, a frequência de visitas dos utilizadores, entre outros.

2.4.3.1 *HiBOp*

O *HiBop* é um protocolo baseado no contexto e que, segundo os seus autores, fornece uma solução geral para a recolha e gestão de qualquer tipo de informação de contexto (Boldrini et al., 2007, Boldrini et al., 2008). Para além disso, mostra como usar essas informações de contexto para modelar o comportamento social dos utilizadores. Este protocolo utiliza a topologia de rede, o histórico de contactos e informações de contexto de utilizadores (tais como endereço de casa, trabalho, etc.) para permitir que um nó aprenda as suas semelhanças com outros nós, permitindo transmitir desta forma uma mensagem entre utilizadores semelhantes. Por outras palavras, um nó que apresente informações semelhantes ao nó de destino tem mais probabilidades de, no futuro, se encontrar com esse destino.

De forma a avaliarem este protocolo, os autores fizeram uma comparação com os protocolos mais populares, nomeadamente o Epidémico e o *Prophet*. Os resultados dessa comparação mostram que o *HiBop* é capaz de reduzir drasticamente o consumo de recursos e a taxa de perda de mensagens, preservando o desempenho em termos do atraso. Contudo, e como referem Mota et al., o estudo não apresenta uma discussão sobre o tempo que é necessário para o protocolo aprender a informação de contexto.

2.4.3.2 *SimBet*

O *SimBet* é um protocolo de encaminhamento que se baseia em interações sociais, explorando métricas de centralidade e de similaridade social de forma a determinar a probabilidade do nó contactar o destino (Daly e Haahr, 2007). Estas métricas de centralidade tem como objetivo medir a importância de um determinado nó dentro da rede, que quando aplicadas ao contexto social permitem identificar quais os nós mais influentes socialmente no seio da rede. Os nós que apresentem valores altos em termos de centralidade servirão de pontes entre as diferentes comunidades da rede. Já os nós que apresentem uma similaridade superior têm maior probabilidade de se encontrarem com nós vizinhos que sejam similares ao destino.

De forma avaliarem o desempenho deste protocolo, os autores utilizaram um *trace* de dados reais, mostrando que o mesmo apresenta um desempenho próximo do protocolo epidémico e com uma redução significativa na sobrecarga. Para além disso, mostram que o *SimBet* supera o protocolo *Prophet*, em particular quando os nós de origem e destino possuem uma baixa conectividade.

2.4.3.3 *BubbleRap*

À semelhança do protocolo *SimBet*, o *BubbleRap também é* um protocolo de encaminhamento baseado em interações sociais, que utiliza igualmente métricas sociais e estruturais para melhorar o desempenho na entrega de mensagens (Hui et al., 2011). Este protocolo explora duas métricas, nomeadamente centralidade e comunidade, utilizando *traces* reais de mobilidade humana. Primeiramente, os autores propõem um método distribuído para detetar comunidades, o que permite a cada nó conhecer a sua comunidade. Depois, utilizam a métrica de centralidade como uma pontuação para cada nó.

Segundo os resultados da avaliação efetuada pelos autores, este protocolo pode melhorar substancialmente o desempenho do encaminhamento de mensagens, em comparação com outros protocolos anteriormente propostos, como é o caso do *Prophet* e do *SimBet*.

Para além destes 3 últimos protocolos apresentados, podem igualmente ser mencionados como protocolos de contexto completo outros protocolos, tais como o SSAR (Li et al., 2010), o *People Rank* (Mtibaa et al., 2010), o SREP (Xie et al., 2011) e o 3R (Vu et al., 2011).

CAPÍTULO 3
ABORDAGEM METODOLÓGICA

De forma a cumprir todos os objetivos inicialmente definidos e criar diretrizes para todo o processo de desenvolvimento foi imprescindível definir uma metodologia que melhor se ajustasse aos objetivos.

O primeiro passo desta metodologia consistiu na tomada de conhecimento da área das redes oportunistas, tentando perceber essencialmente o seu princípio de funcionamento, os estudos que já foram feitos e os avanços científicos que advieram dos mesmos e as técnicas que têm sido usadas para avaliar o desempenho destas redes. É com base nesta fase inicial que se consegue perceber se os objetivos definidos inicialmente permitem contribuir cientificamente para a área.

Posteriormente a estas etapas iniciais, deu-se início ao desenho de algoritmos adequados à avaliação de desempenho destas redes com base em *traces* de redes Wi-Fi. Numa primeira fase pretendia-se gerar dados fictícios através de uma ferramenta desenvolvida para o efeito, extrair encontros a partir desses dados e encaminhar mensagens durante esses encontros. Para tal foram desenvolvidos três algoritmos:

- **Algoritmo para geração de *logs* RADIUS sintéticos**

  Este algoritmo permitiu gerar dados fictícios, através da definição de um conjunto de parâmetros de entrada. Estes dados são gravados para uma base de dados e servem de base aos dois algoritmos seguintes. Muito embora o objetivo deste trabalho seja a avaliação do desempenho de redes oportunistas com base em *traces* reais, a possibilidade de geração de *traces* sintéticos contribui para o desenho, implementação e teste, em ambiente controlado, dos restantes algoritmos.

- **Algoritmo para extração de encontros**

  Através deste algoritmo, que tem como entrada os registos gerados do algoritmo anterior, ou traces reais, foram extraídos encontros entre as diversas stations existentes na rede. Estes encontros são igualmente gravados numa base de dados.

- **Algoritmo para o encaminhamento epidémico de mensagens**

  Este terceiro algoritmo permitiu que as stations gerassem mensagens para a rede e que consequentemente fossem encaminhadas epidemicamente durante os encontros extraídos no algoritmo anterior. No fim, é possível extraírem-se resultados suscetíveis de análise, de forma avaliar o desempenho desta rede com dados fictícios. Para isso foram identificadas e aplicadas determinadas métricas de desempenho que são caracterizadas mais à frente.

Como se poderá verificar no capítulo seguinte, após o desenho e implementação de cada um destes três algoritmos, os mesmos foram submetidos a dois tipos de testes: de correção e de desempenho.

Ao contrário desta primeira fase que engloba somente dados sintéticos, na segunda fase, caso haja tempo, pretende-se utilizar dados reais, executando somente os dois últimos algoritmos, ou seja, extraírem-se encontros através desses dados e encaminharem-se mensagens durante os encontros.

Seguidamente é descrita com maior detalhe toda a metodologia utilizada no desenvolvimento deste trabalho, desde a caracterização dos dados utilizados até às escolhas tecnológicas.

3.1 Caracterização dos dados

A larga disponibilização de registos sobre a utilização de redes sem fios tem permitido a realização de vários estudos, como forma de apurar o possível desempenho de redes oportunistas. Os dados utilizados ao longo desta dissertação inserem-se nessa perspetiva, tendo sido especificamente caracterizados em dois tipos, dados sintéticos e dados reais. Os primeiros referem-se a *logs* RADIUS sintéticos gerados por um gerador concebido exclusivamente para o propósito desta dissertação, enquanto que os segundos correspondem a *logs* RADIUS recolhidos num ambiente real específico. De seguida, são caracterizados mais detalhadamente cada um desse conjunto de dados que foram utilizados.

3.1.1 Dados Sintéticos

Como foi referido anteriormente, a primeira tarefa de desenvolvimento correspondeu ao desenho e conceção de um algoritmo capaz de gerar *logs* RADIUS sintéticos, de forma a serem utilizados nos dois algoritmos seguintes. A geração desses *logs* ou registos é feita de acordo com cinco parâmetros de entrada, nomeadamente o tempo de simulação, a área de simulação, o número de stations, o número de APs e a área de cobertura de cada um deles. No início de execução do algoritmo são atribuídos manualmente valores para cada um desses parâmetros, permitindo com isso criar ambientes de simulação distintos e perfeitamente moldáveis.

Especificamente para o desenvolvimento deste trabalho, foram utilizados 3852 registos sintéticos, gerados através desse primeiro algoritmo, sendo utilizados maioritariamente para os testes de correção dos algoritmos. Para os testes de desempenho foram gerados diferentes conjuntos de registos sintéticos, de forma a testar o desempenho dos algoritmos em diferentes ambientes. Os 3852 registos foram gerados através da seguinte configuração inicial:

Área de simulação: 256 m^2

Número de APs: 5

Raio de Cobertura de cada AP: 20

Número de Stations: 100

Tempo de simulação: 1000s

Esta configuração é igualmente apresentada na tabela 4.2 do próximo capítulo.

Cada um dos registos gerados e escritos na base de dados possui o formato que é ilustrado na figura seguinte.

t	StationID	ApID	EventType
34	1	6	STOP

Figura 3.1 – Exemplo representativo de um registo sintético presente na base de dados

O campo t corresponde ao instante de tempo em que é gerado o registo, ou seja, quando uma dada station entra ou sai na área de cobertura de um determinado AP. Como é percetível o campo StationID e ApID referem-se, respetivamente, ao id da station e do AP envolvidos. O último campo, denominado EventType, corresponde ao tipo de evento que ocorreu, podendo ser do tipo Start (a station entrou na área de cobertura de um AP) ou do tipo Stop (a station saiu da área de cobertura de um AP). Seguindo o exemplo ilustrado na figura anterior, no instante 34, a station1 saiu do AP6 e com isso ocorreu um evento do tipo Stop e consequentemente gerado um registo com todas estas características registadas.

Para além dos campos ilustrados na figura 3.1, foi adicionado igualmente na base de dados e de forma automaticamente incremental, um campo referente ao id de cada registo, que se inicia em 1. Este campo permite identificar facilmente qualquer registo, sendo útil em qualquer acesso e manipulação de dados necessário.

3.1.2 Dados Reais

Relativamente aos dados reais, os mesmos foram recolhidos no âmbito do projeto SUM (*Sensing and Understanding human Motion dynamics*) (Moreira et al., 2012). Estes dados descrevem o movimento de pessoas utilizando as redes sem fios *eduroam*, explorando os *logs* do serviço RADIUS.

Os *logs* RADIUS deste projeto foram recolhidos em 5 diferentes *campus* universitários, Universidade do Minho (UM), *campus* de Azurém e Gualtar, Universidade do Porto (UP), Universidade de Lisboa (UL) e Universidade de Coimbra (UC) e durante um período total de 22 meses, mais especificamente de 2010 a 2012. A tabela seguinte, adaptada da presente no relatório técnico referido anteriormente, resume o número de registos catalogados em cada *campus* e em que períodos de tempo.

Tabela 3.1 – Resumo dos dados reais catalogados.

Fonte	Inicio	Fim	Número de registos	Total de registos
UM	05.2010	12.2010	16.229.182	16.229.182
UM	01.2011	12.2011	15.892.009	32.121.191
UM	01.2012	06.2012	18.314.304	50.435.495
UP	05.2010	12.2010	1.757.996	52.193.491
UP	01.2011	05.2011	1.545.521	53.739.012
UL	04.2011	12.2011	13.964.268	67.703.280
UL	01.2012	11.2012	12.061.736	79.765.016
UC	05.2010	12.2010	6.037.662	85.802.678
UC	01.2011	12.2011	13.373.649	99.176.327
UC	01.2012	05.2012	8.144.580	107.320.907

3.2 Identificação e caracterização das métricas de desempenho de redes oportunistas

Existem diferentes formas de avaliar o desempenho de uma rede oportunista. É difícil perceber quais destas formas ou métricas irão produzir resultados mais úteis e ajustados, antes das mesmas serem registadas e analisadas. Isto acontece pois existem métricas que apresentam um elevado índice de variação em determinados ambientes ou circunstâncias. Por exemplo, o tempo que uma mensagem leva para que seja recebida por todos os nós de uma rede pode variar muito devido a alguns nós poderem residir em locais particularmente difíceis de alcançar (Coombs, 2014).

Assim, para este trabalho foram identificadas e aplicadas quatro principais métricas de desempenho. Focou-se somente nestas quatro métricas com o intuito de aumentar a probabilidade de se produzir uma avaliação acertada e útil. Seguidamente são apresentadas as métricas utilizadas.

- **Atraso:** corresponde à duração entre o tempo de geração de uma mensagem e o tempo de entrega da mesma ao seu destino.

  Para extrair o atraso no encaminhamento das mensagens foi necessário registar dois valores temporais em dois momentos diferentes durante o processamento do algoritmo. O primeiro aquando da geração de cada mensagem e o segundo aquando da sua entrega. Desta forma, a diferença entre estes dois valores permite registar o atraso de comunicação de cada mensagem.

  Atraso = Tempo de entrega – Tempo de geração

- **Taxa de entrega:** corresponde à razão entre o número total de mensagens entregues ao seu destino e o total de mensagens geradas pelos nós de origem.

  Para extrair esta métrica foi necessário registar igualmente dois valores durante o processamento do algoritmo, nomeadamente o número de mensagens que foram geradas e o número de mensagens que efetivamente chegaram ao seu destino.

  Taxa de entrega = (Total de mensagens entregues) / (Total de mensagens geradas)

- **Comprimento médio da fila de espera:** corresponde à média do número de mensagens que cada station possui na sua fila ao longo do tempo, calculado para todas as stations.

- **Número de mensagens transmitidas em cada station:** corresponde ao número de mensagens que são transmitidas por cada station ao longo da simulação. Desta forma é possível verificar a energia consumida por cada station, pois uma vez que as stations são nós móveis e este nós por norma são alimentados por baterias, uma station que transmita muitas mensagens fica mais facilmente sem energia na sua bateria.

3.3 Especificação/Desenho do sistema

Seguidamente é ilustrado todo o processo de desenvolvimento do sistema.

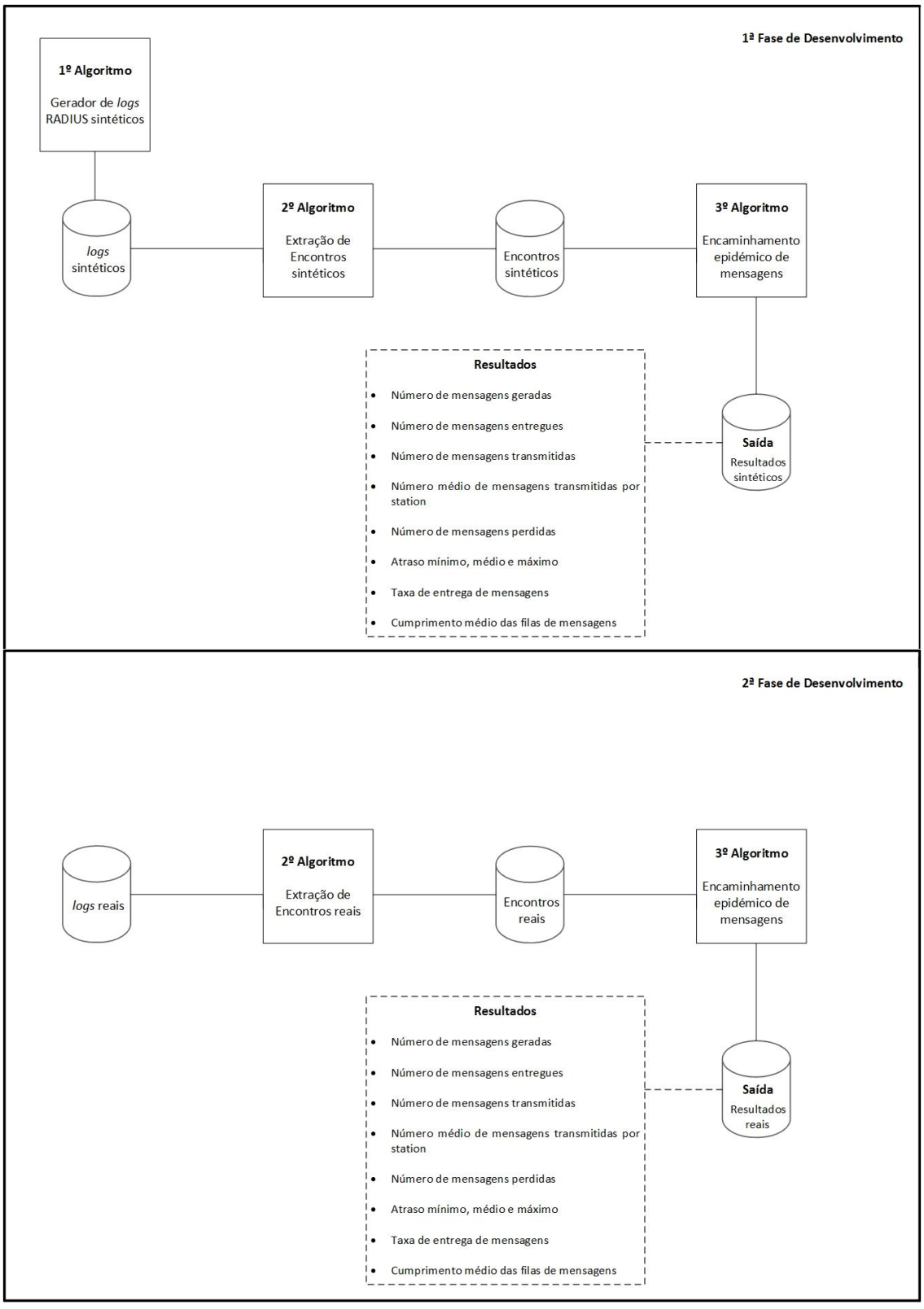


Figura 3.2 – Desenho do sistema

3.4 Escolha de tecnologias

De forma a ir de encontro aos objetivos delineados inicialmente, era imprescindível a escolha de tecnologias que se ajustassem ao que era pretendido. É importante referir que más escolhas tecnológicas podiam levar a atrasos consideráveis no decorrer do trabalho, entre outras contrariedades. Assim, de forma a evitar estes contratempos, dividiu-se essencialmente esta escolha em três partes principais, uma primeira referente à linguagem de programação a utilizar, uma segunda relativa à base de dados e uma terceira relativa ao ambiente computacional utilizado ao longo do trabalho. Seguidamente é feita uma descrição mais pormenorizada das duas primeiras escolhas e os motivos para tais decisões e no final é apresentada uma listagem referente ao ambiente computacional utilizado.

3.4.1 Linguagem de Programação

Como foi referido anteriormente, um dos objetivos deste trabalho era combinar as capacidades de processamento dos sistemas de gestão de base de dados com linguagens de programação orientadas a objetos, de forma a desenvolver ferramentas eficientes de processamento de dados. Face a isto, o primeiro ponto considerado na escolha da linguagem de programação foi o paradigma de programação orientadas a objetos.

Este paradigma introduz uma série de novas abordagens na programação, como por exemplo, atributos e classes, otimizando o desenvolvimento de sistemas. Para além dos termos referidos anteriormente, e como o próprio nome indica, a definição de objeto é essencial ao paradigma da POO. O conceito de objeto pretende concentrar em si todas as virtudes de um modelo de desenvolvimento de software que se baseia nas seguintes propriedades [Martins, 2009]:

- A independência de contexto (que permite reutilização);
- A abstração de dados (que garante abstração);
- O encapsulamento (que garante abstração e proteção);
- A modularidade (que garante composição por partes).

Um objeto é uma entidade que deve ocultar do exterior a sua estrutura interna, mostrando apenas um conjunto de operações que é capaz de executar quando são externamente invocadas (Martins, 2009).

Assim, a escolha da linguagem de programação recaiu no JAVA. Optou-se por esta linguagem por um conjunto de razões. Primeiro, porque é uma das linguagens mais utilizadas para programar usando objetos (e não só), tornando-se numa escolha lógica; Segundo, por ser uma linguagem de alto nível, que aliada ao paradigma da POO faz desta linguagem uma boa escolha para desenvolvimento; Terceiro, pela boa documentação, grande comunidade e suporte que possui, não esquecendo o facto de ser gratuito e de código aberto; Quarto, pela possibilidade de executar o seu código em diferentes ambientes, tendo como principal característica a filosofia "Escreva uma Vez e Execute em Qualquer Lugar"; Quinto, pelo facto da interface JDBC fazer com que a comunicação com a base de dados funcione de forma homogénea; entre outras.

3.4.2 Base de dados

Relativamente ao sistema de gestão de base de dados escolheu-se o MySQL. À semelhança da escolha da linguagem de programação, a escolha relativa à base de dados também foi motivada por algumas razões. De entre várias razões, destaca-se o facto do MySQL ter um excelente desempenho e estabilidade, ser de fácil utilização e de possuir um suporte alargado, face à popularidade do mesmo. Para além disso, outra razão que pesou na escolha foi a experiência pessoal, pois já foi utilizado noutros trabalhos académicos.

Para ser possível efetuar a ligação entre o JAVA e a base de dados, enviar instruções SQL e representar o resultado através de uma estrutura de dados própria, utiliza-se o JDBC. É através deste conjunto de ações padrão que é permitido a um programa JAVA interagir com uma base de dados. Esta ligação é descrita com maior detalhe no próximo capítulo.

De forma a integrar facilmente algumas tecnologias e permitir um ambiente de simulação adequado, utilizou-se o XAMPP. Esta distribuição gratuita e de fácil configuração permitiu instalar e integrar de uma só vez a base de dados MySQL e o Apache.

Em termos de base de dados e como pode ser visto na figura 3.2, é necessário gravar na mesma os seguintes conteúdos:

- *Logs* sintéticos gerados no primeiro algoritmo;
- Encontros extraídos dos *logs* sintéticos gerados anteriormente, sendo esta extração feita no segundo algoritmo;
- Resultados sintéticos resultantes do encaminhamento epidémico de mensagens que foi feito no terceiro algoritmo;
- *Logs* de *traces* reais;

- Encontros extraídos dos *logs* reais considerados anteriormente, sendo esta extração igualmente feita através do segundo algoritmo;
- Resultados reais resultantes igualmente do encaminhamento epidémico de mensagens que foi feito no terceiro algoritmo;

3.4.3 Ambiente computacional

A listagem apresentada seguidamente corresponde a todo o ambiente computacional utilizado ao longo de todo o trabalho, nomeadamente na implementação dos algoritmos desenvolvidos e nos respetivos testes aos mesmos.

Hardware:

Processador: Intel Core i7 3630QM CPU 2.40GHZ

Memória RAM: 8GB

Software:

Sistema Operativo: Windows 8.1 x64 / Windows 10 x64

Ambiente de desenvolvimento: Eclipse Luna SR1a 4.4.1

Versão do Java: 8

Versão do XAMPP: 3.2.1

Versão do MySQL: 5.5.32

Versão do Apache: 2.4.4

CAPÍTULO 4
DESENVOLVIMENTO DE ALGORITMOS

Este capítulo corresponde a todo o processo de desenvolvimento dos três algoritmos, estando dividido em três subcapítulos distintos, correspondendo a cada um dos algoritmos. Para cada um dos subcapítulos são apresentados todos os passos e decisões tomadas no desenvolvimento do algoritmo, a sua implementação e por fim os testes ao mesmo (correção e desempenho).

4.1. Algoritmo para geração de *logs* RADIUS sintéticos

O primeiro algoritmo desenvolvido tem como propósito gerar *logs* RADIUS sintéticos. Para tal foram definidos no algoritmo vários parâmetros principais de simulação que servem de entrada: o número de APs, o raio de cobertura de cada um deles, o número de *stations*, o tempo de simulação, as medidas da área de simulação (Lx e Ly), o tempo máximo para um movimento aleatório de cada station (tmax), a velocidade máxima a que uma station pode deslocar-se (vmax) e o instante inicial de cada movimento de cada station (t0).

Tanto a posição dos APs como das stations são inicializadas na área de simulação de forma aleatória. Logicamente, os APs ao serem inicializados permanecem na mesma posição durante toda a simulação. Contrariamente aos APs, as stations movimentam-se aleatoriamente durante todo o tempo de simulação. Primeiramente é calculado um tempo aleatório para a station executar o movimento, sendo esse tempo definido através da seguinte expressão:

$$St\,[k] + \Delta t\ (random)$$

,ou seja, é a soma do instante de tempo corrente com um tempo aleatório compreendido no intervalo [t0, tmax]. Este intervalo é definido pelo instante inicial que é zero (t0) e o tempo máximo definido para um movimento (tmax).

Seguidamente é calculada aleatoriamente a nova posição da station decorrente do seu movimento. Esta posição é calculada através da seguinte expressão:

$$Sp\,[k] + \Delta d\ (random)$$

,ou seja, é a soma da posição atual da station com uma distância aleatória compreendida no intervalo [-vmax * Δt, vmax * Δt]. Sendo a posição da station uma grandeza vetorial (x,y), esta distância aleatória representada por $\Delta d\ (random)$ também o é e por isso o seu x e o seu y são definidos da seguinte forma:

$$x = xcurrent + (randomdouble - 0.5) * vmax * \Delta t\ (random)$$

$$y = ycurrent + (randomdouble - 0.5) * vmax * \Delta t\ (random)$$

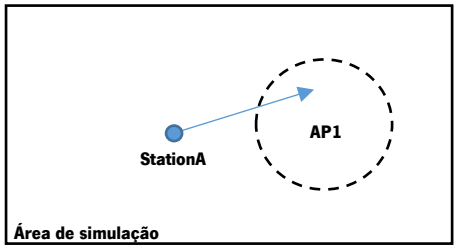
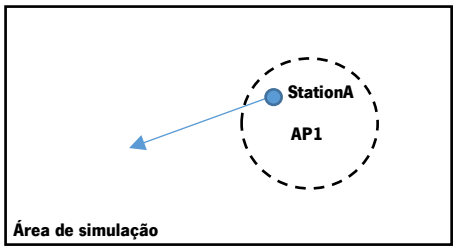
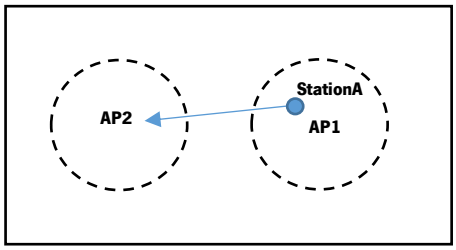
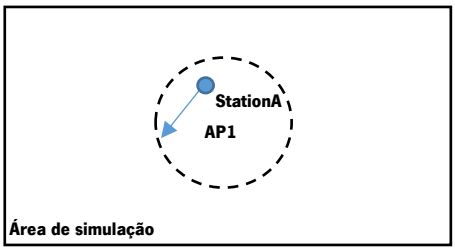
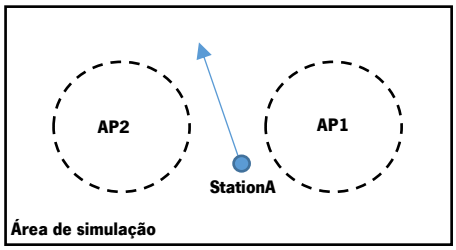
, em que xcurrent e ycurrent são as posições atuais do x e y e randomdouble é um valor aleatório compreendido entre 0 e 1.

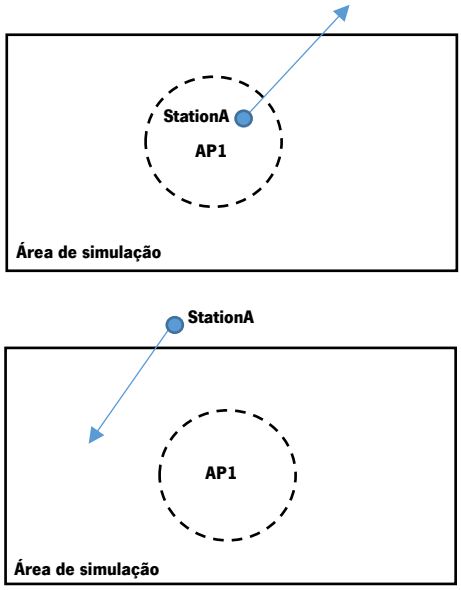
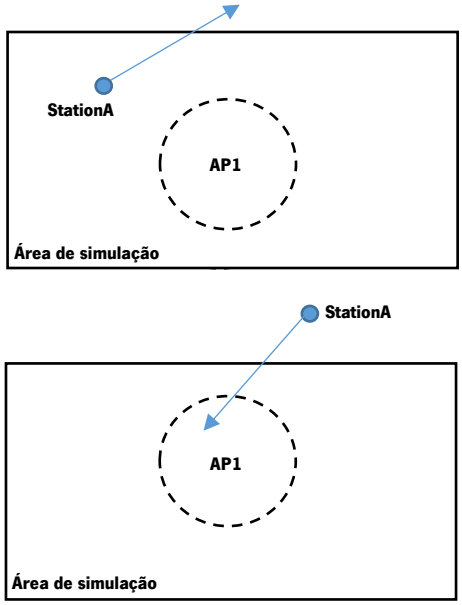
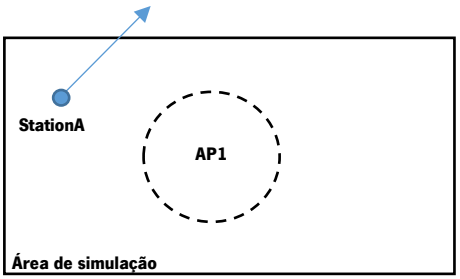
As stations movem-se aleatoriamente desta forma até o tempo definido para a duração da simulação chegar ao fim. Durante este tempo, caso se movimentem para fora da área de simulação são logo de seguida reinseridas na mesma, exatamente do mesmo modo que são inicializadas no início da simulação, ou seja dentro da área de simulação e de forma aleatória.

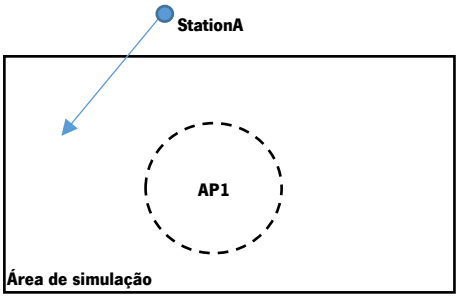
Como pode ser visto na figura 4.1, onde é apresentada a descrição do algoritmo, o *output* ou saída do mesmo são registos. Estes registos são principalmente caraterizados pelo seu tipo, ou seja, os mesmos poderão ser de dois tipos diferentes, *Start* ou *Stop*. Esta classificação está particularmente relacionada com o movimento aleatório de cada station e com a posição de cada AP, uma vez que apenas são gerados registos quando as stations entram ou saem da área de cobertura dos APs. É gerado um registo *Start* sempre que uma station entra na área de cobertura de um determinado AP e um registo Stop caso aconteça o inverso, ou seja, caso uma station saia da área de cobertura de um dado AP.

Assim, o movimento aleatório de cada station e a consequente classificação de cada registo é ilustrada na tabela seguinte.

Tabela 4.1 – Possíveis movimentos aleatórios de cada station e classificação dos registos gerados.

Movimento da station	Descrição do movimento	Registos gerados
	A StationA move-se para dentro da área de cobertura do AP1.	É gerado um registo **START** associado ao AP1**.**
	A StationA move-se para fora da área de cobertura do AP1.	É gerado um registo **STOP** associado ao AP1.
	A StationA sai da área de cobertura do AP1 diretamente para dentro da área de cobertura do AP2.	É gerado um registo **STOP** associado ao AP1 e um registo **START** associado ao AP2.
	A StationA movimenta-se dentro da área de cobertura do AP1.	Não são gerados registos.
	A StationA movimenta-se fora da área de cobertura de qualquer AP.	Não são gerados registos.

	A StationA movimenta-se de dentro da área de cobertura do AP1 para fora da área de simulação. A StationA é colocada de novo aleatoriamente dentro da área de simulação e não fica posicionada dentro da área de cobertura de qualquer AP.	É gerado um registo **STOP** associado ao AP1, quando a StationA sai da área de cobertura do mesmo.
	A StationA movimenta-se para fora da área de simulação sem estar dentro da área de cobertura de qualquer AP. A StationA é colocada de novo aleatoriamente dentro da área de simulação e na área de cobertura do AP1.	É gerado um registo **START** associado ao AP1, quando a StationA entra na área de cobertura do mesmo.
	A StationA movimenta-se para fora da área de simulação sem estar dentro da área de cobertura de qualquer AP. A StationA é colocada de novo	Não são gerados registos

	aleatoriamente dentro da área de simulação e não fica posicionada dentro da área de cobertura de qualquer AP.

Cada um dos registos gerados é descrito através de 4 campos:

- **t:** instante de tempo da geração do registo, ou seja, corresponde ao instante de tempo da entrada ou saída da station da área de cobertura do AP;
- **stationID**: identificação da station que está a ser processada;
- **apID:** identificação do AP ao qual a station está associada no instante t;
- **eventType:** tipo de evento gerado, Start ou Stop.

Seguidamente à geração dos registos, os mesmos são escritos numa base de dados de forma a servirem de suporte aos algoritmos e aos diferentes processos de análise realizados posteriormente. No final, os registos gerados e enviados para a base de dados terão o seguinte formato:

t, stationID, apID, eventType

De seguida apresenta-se toda a descrição deste primeiro algoritmo sob a forma de pseudocódigo.

Algoritmo para gerar logs RADIUS sintéticos

St [k], Sp [k] e Sap [k]: posição das stations;
APs [j]: posição dos APs;
D [j]: distância da station a cada AP;
R: raio de cobertura do AP;
t0: instante de tempo inicial relativo ao movimento da station;
tf: instante de tempo final relativo ao movimento da station;
tmax: tempo máximo para cada movimento aleatório da station;
vmax: velocidade máxima da station em cada movimento aleatório;

Entrada: número de APs, número de stations, R, tmax, vmax, t0, tf, Lx e Ly.
Saída: *logs RADIUS* sintéticosno formato: t, StationID, ApID, EventType.

```
1.  Início;
2.  Distribuir aleatoriamente os APs dentro da área de simulação, sendo essa distribuição uniforme;
3.  Para cada station de k:
4.      St [k] ← t0;
5.      Sp [k] ← random dentro da área de simulação;
6.      Sap [k] ← APnearest, se d<R ou null se d>R;
7.  Fim Para;
8.  Para cada station de k:
9.      i ← 0;
10.     Enquanto St [k] ≤ tf
11.         St [k] ← St [k] + Δt (random);
12.         Sp [k] ← Sp [k] + Δd (random);
13.         Para cada Apj:
14.             D [j] ← (√(xk – xj) * (xk – xj) + (yk – yj) * (yk – yj), j), em (xk,yk) são as coordenadas das
15.                     stations e (xj,yj) são as coordenadas dos APs;
16.         Fim Para;
17.         Ordenar D [j] por ordem crescente de distância;
18.         d ← D[0];
19.         APnearest ← APj;
20.         Verificar associação a AP e gerar registo:
21.          Se d<R e Sap [k] = null
22.             Sr [k,i] ← gerar START;
23.         Senão d<R e Sap [k] ≠ APnearest
24.             Sr [k,i] ← gerar STOP;
25.             Sr [k,i] ← gerar START;
26.          Senão d>R e Sap [k] ≠ null
27.             Sr [k,i] ← gerar STOP;
28.         Fim Se;
29.         i ← i+1;
30.     Fim Enquanto;
31.     Escrever registos gerados para a base de dados;
32. Fim Para;
33. Fim.
```

Figura 4.1 – Pseudocódigo do algoritmo para geração de logs RADIUS sintéticos

4.1.1 Implementação

Após o desenvolvimento do algoritmo para geração de registos RADIUS sintéticos, o próximo objetivo passava pela implementação do mesmo na linguagem de programação orientada a objetos escolhida, o JAVA.

Em JAVA, as classes geralmente são agrupadas em *packages*, sendo assim possível manter as classes em compartimentos ou pacotes distintos, agrupadas em função da sua funcionalidade. Em especial, tal facto facilita a procura de classes, dado que os *packages* têm nomes bastante representativos da funcionalidade das classes que os constituem [Martins, 2009]. Assim, a implementação deste algoritmo foi divida em dois *packages*, um referente à base de dados, denominado *database* e outro referente a todo o processo de gerar os registos, denominado *generator*.

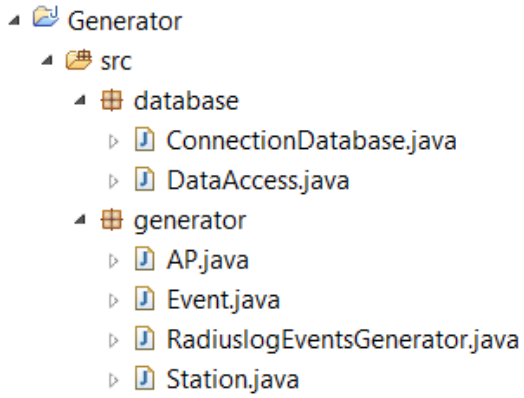


Figura 4.2 – Organização da implementação do algoritmo para geração de registos RADIUS sintéticos

No anexo 1 é apresentada toda a implementação deste algoritmo de forma mais detalhada.

4.1.2 Testes

Após a tarefa de implementação estar concluída, é imprescindível submeter a mesma a vários testes. Para tal, optou-se por dividi-los em dois tipos, testes de correção e testes de desempenho.

De forma a ir ao encontro de simulações com valores realistas, em ambos os tipos de testes, considerou-se o número de APs em função da área de simulação e o número de stations em função do número de APs. Assim, foi considerado a existência de um AP por cada 50m^2 de área e 20 stations por cada AP. Relativamente ao tempo de simulação (tf) foram considerados 4

diferentes valores, 10, 100,1000 e 10000 segundos, de forma a perceber se os resultados tempo de execução e número de registos gerados cresce linearmente com o tempo de simulação.

4.1.2.1 Testes de Correção

A. Parâmetros considerados

Para serem executados os testes de correção neste primeiro algoritmo foram considerados os parâmetros presentes na tabela que se segue.

Tabela 4.2 – Parâmetros considerados nos testes de correção do algoritmo para geração de logs RADIUS sintéticos.

Tipo de parâmetro	Parâmetro	Valor
Simulação/Entrada	Área de simulação	256 m^2
	Número de APs	5
	Raio de cobertura de cada AP	20
	Número de Stations	100
	Tempo de simulação	1000s
Saída	Registos	———
Análise	Formato e lógica do registo	———

B. Resultados e Análise

Configurado todo o cenário de simulação, foi executado o algoritmo e no final da sua execução foram gerados 3852 registos.

De forma a analisar corretamente os dados, foi feita uma análise de todos os registos e verificou-se que os mesmos foram gerados corretamente, ou seja, no formato t, StationID, ApID, EventType. Já na base de dados, para além destes campos foi adicionado e incrementado automaticamente o campo relativo ao id.

Analisando com mais detalhe um conjunto de registos referentes a uma única station, neste caso a $station_0$, pode confirmar-se o correto formato dos registos. A figura seguinte, correspondente a 10 registos dos 50 gerados relativos à $station_0$, ilustra isso mesmo.

id	t	StationID	ApID	EventType
1	6	0	2	STOP
2	6	0	3	START
3	56	0	3	STOP
4	56	0	4	START
5	168	0	4	STOP
6	168	0	1	START
7	191	0	1	STOP
8	191	0	4	START
9	239	0	4	STOP
10	239	0	3	START

Figura 4.3 – Resultados: amostra de 10 registos gerados para a base de dados referente à station0 durante os testes de correção do algoritmo para geração de logs RADIUS sintéticos.

Como se pode verificar na figura anterior, a station0 move-se aleatoriamente no conjunto de registos selecionado entre quatro APs: 1, 2, 3 e 4. Para entender melhor o comportamento da station0 descreve-se seguidamente cada um dos registos.

1|6, 0, 2, STOP

2|6, 0, 3, START: A $station_0$ entra na área de cobertura do AP3 (gera um registo de START neste AP) e com isso gera também um registo de STOP no AP que estava ligado anteriormente, que neste caso é o AP2 (registo anterior, id = 1). Neste caso a $station_0$ saiu da área de cobertura do AP2 diretamente para a área de cobertura do AP3);

3|56, 0, 3, STOP

4|56, 0, 4, START: Após 50s a $station_0$ sai da área de cobertura do AP3 diretamente para a área de cobertura do AP4, gerando novamente dois registos, um registo START no AP que entrou (AP4) e um registo STOP associado ao AP que saiu, o AP3 (registo com id = 3);

5|168, 0, 4, STOP

6|168, 0, 1, START: Após 112s a $station_0$ sai da área de cobertura do AP4 diretamente para a área de cobertura do AP1, gerando novamente dois registos, um registo START no AP que entrou (AP1) e um registo STOP associado ao AP que saiu, o AP4 (registo com id = 5);

7|191, 0, 1, STOP

8|191, 0, 4, START: Após 23s a $station_0$ sai da área de cobertura do AP1 e move-se novamente para dentro da área de cobertura do AP4, gerando com isso um registo STOP associado ao AP que saiu (AP1, registo com id = 7) e um registo START associado ao AP em que entra (AP4, presente registo com id = 8);

9|239, 0, 4, STOP

10|239, 0, 3, START: Após 49s a $station_0$ sai da área de cobertura do AP4 diretamente para a área de cobertura do AP3, gerando novamente dois registos, um registo START no AP que entrou (AP3) e um registo STOP associado ao AP que saiu, o AP4 (registo com id = 9).

Constata-se então que os registos foram gerados corretamente e a $station_0$ apresenta um comportamento realista. Nos registos selecionados para análise, a $station_0$ apresentou sempre o movimento de sair da área cobertura de um AP diretamente para a área de cobertura de um outro AP. Este comportamento vai ao encontro dos possíveis comportamentos de uma station durante o seu movimento aleatório, ilustrados anteriormente na tabela 4.1

4.1.2.2 Testes de Desempenho

A. Parâmetros considerados

Após a implementação e a execução dos testes de correção estarem concluídos, o próximo passo passava por testar o desempenho do algoritmo.

De forma a avaliar ao máximo o desempenho deste algoritmo, foram executados dois conjuntos de testes de desempenho diferentes. O primeiro conjunto de testes foi executado utilizando a mesma configuração de simulação em todos eles, variando somente o tempo de simulação. Executaram-se 5 simulações para cada tempo de simulação (10s, 100s, 1000s e 10000s), de forma avaliar a linearidade do tempo de execução do algoritmo e do número de registos gerados entre cada uma das 5 simulações de cada tempo específico. Assim, nestes primeiros testes definiu-se a seguinte configuração de simulação:

- **Área de simulação:** 1024 m^2
- **Número de APs:** 20
- **Raio de cobertura de cada AP:** 20m
- **Número de Stations:** 400

. Já para os segundos testes, executaram-se 10 simulações onde foram mantidos em todas simulações o valor de tempo de simulação (1000s) e o raio de cobertura de cada AP (20m), variando os restantes parâmetros. Assim, nestes segundos testes definiu-se a seguinte configuração de simulação

- **Área de simulação:** 256m^2, 484m^2, 729m^2, 1024m^2, 1296m^2, 1521m^2, 1769m^2, 2025m^2, 2304m^2 e 2500m^2
- **Número de APs:** 5, 10, 15, 20, 25, 30, 35, 40, 45 e 50
- **Raio de cobertura de cada AP:** 20m
- **Número de Stations:** 100, 200, 300, 400, 500, 600, 700, 800, 900 e 1000

Aquando a apresentação dos resultados destes testes são igualmente salientados os valores utilizados em cada uma das simulações executadas.

A tabela seguinte resume os parâmetros gerais considerados nestes testes, apresentando os intervalos de valores utilizados.

Tabela 4.3 – Parâmetros considerados nos testes de desempenho do algoritmo para geração de *logs* RADIUS sintéticos.

Tipo de parâmetro	Parâmetro	Valor
Simulação/Entrada	Área de simulação	$256m^2$ - $1024m^2$
	Número de APs	5 - 50
	Raio de cobertura de cada AP	20m
	Número de Stations	100 - 1000
	Tempo de simulação	10, 100, 1000 e 10000s
Saída	Registos	————
Análise	Tempo de execução	————
	Número de registos gerados	————

B. Resultados e Análise

Após todo o ambiente de simulação estar configurado, deu-se início aos primeiros testes de desempenho deste primeiro algoritmo.

A tabela seguinte apresenta os resultados alcançados nos 20 testes executados nestes primeiros testes.

Tabela 4.4 – Resultados dos primeiros testes de desempenho do algoritmo para geração de *logs* RADIUS sintéticos

Tempo de Simulação (s)	Tempo de execução (s)	Número de registos gerados
10	0,22	42
	0,22	38
	0,31	22
	0,31	54
	0,22	34
100	0,6	1758
	0,72	1694
	0,65	1790
	0,59	1762
	0,59	1764
1000	3,26	20272
	3,12	20242
	3,13	20520
	2,95	19660
	3,32	20366
10000	26,28	201686
	29,32	208060
	26,38	202238
	28,1	205946
	28,45	207492

Como foi dito anteriormente, o tempo de execução presente na tabela anterior refere-se ao tempo em que o algoritmo é executado, incluindo com isso não só a geração dos registos mas também a sua escrita na base de dados.

Para uma análise mais detalhada, ilustraram-se os resultados apresentados na tabela anterior em forma de gráfico. O primeiro gráfico apresentado seguidamente representa a evolução do número de registos gerados à medida que é aumentado o tempo de simulação. São representadas as 5 simulações em cada tempo de simulação respetivo, de forma a avaliar a linearidade das mesmas, onde cada barra no eixo do tempo de simulação representa uma dessas simulações. O eixo vertical referente ao número de registos gerados apresenta valores logarítmicos de base 10 de forma a ir ao encontros dos valores apresentados no eixo horizontal, referente ao tempo de simulação.

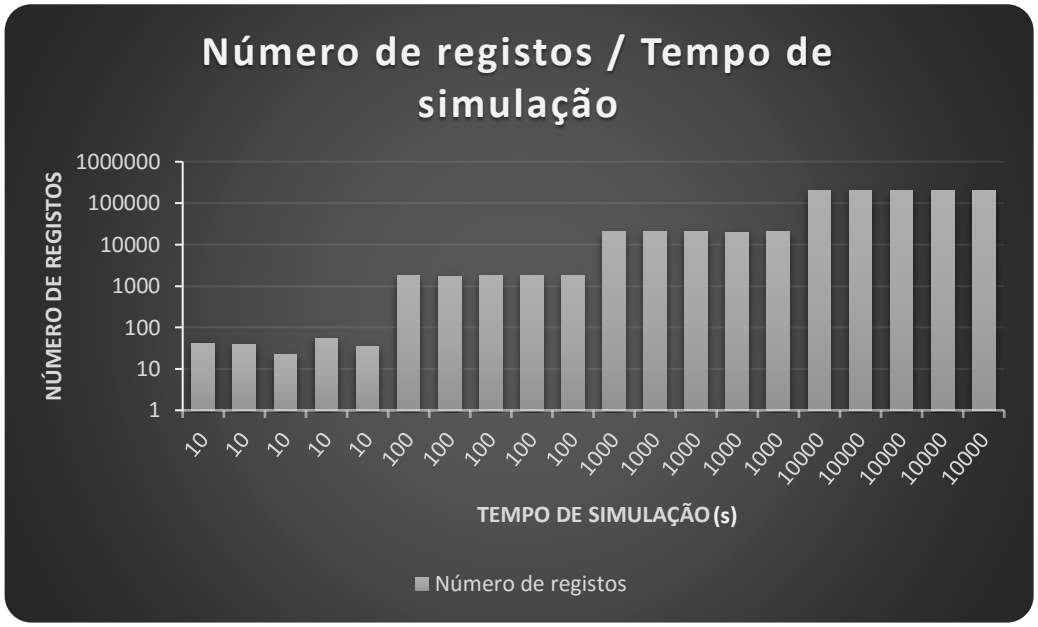


Figura 4.4 – Resultados dos primeiros testes de desempenho do algoritmo para geração de *logs* RADIUS (Número de registos/Tempo de simulação).

Através da visualização do gráfico anterior, é possível verificar a existência de uma estabilidade bem definida no número de registos gerados nas 5 simulações de cada tempo de simulação. Apenas nos 10s é que se constata uma menor estabilidade, o que é normal, uma vez que o tempo de simulação é muito reduzido.

De igual forma, seguidamente apresenta-se um gráfico semelhante ao anterior mas desta vez representando a evolução do tempo de execução do algoritmo à medida que se aumenta o tempo de simulação, ao invés do número de registos gerados.

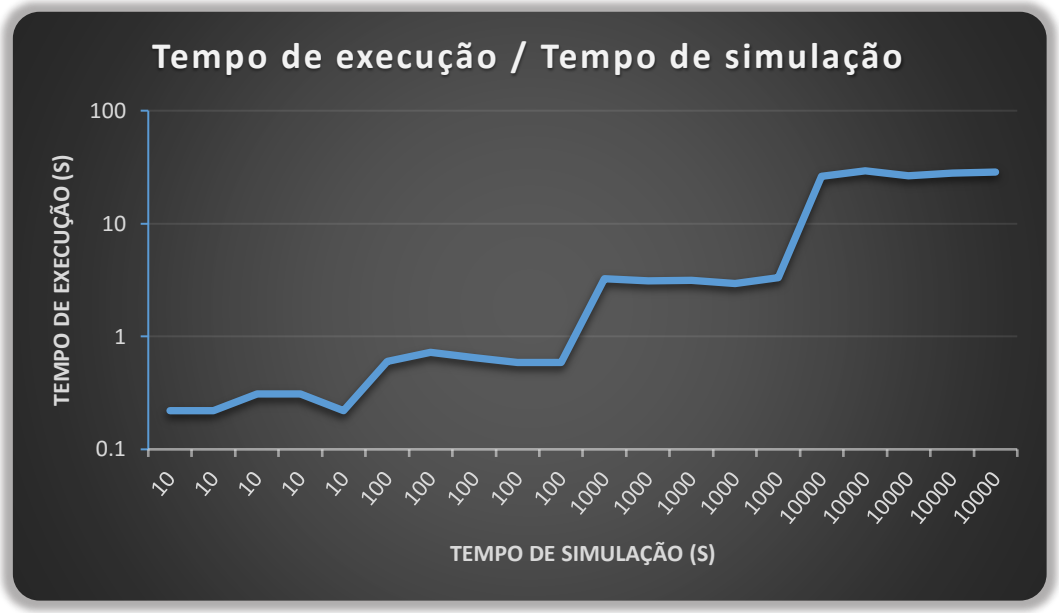


Figura 4.5 – Resultados dos primeiros testes de desempenho do algoritmo para geração de *logs* RADIUS (Tempo de execução/Tempo de simulação).

Como é possível verificar no gráfico anterior, o tempo de execução apresenta uma estabilidade semelhante à alcançada com o número de registos, ou seja uma estabilidade bem definida nas 5 simulações de cada tempo de simulação, com exceção nos 10s.

Seguidamente são apresentados os dois gráficos anteriores numa única representação gráfica.

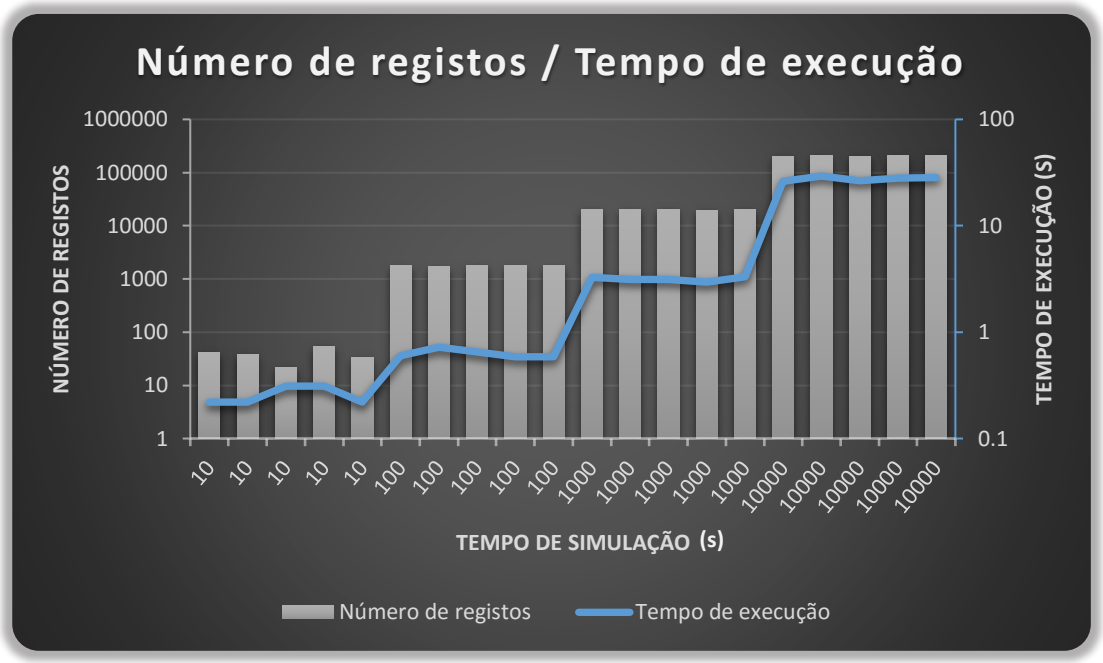


Figura 4.6 – Resultados dos primeiros testes de desempenho do algoritmo para geração de *logs* RADIUS (Número de registos e Tempo de execução/Tempo de simulação).

Como se pode observar no gráfico anterior, a evolução do número de registos gerados é acompanhada paralelamente com a evolução do tempo de execução do algoritmo, isto à medida que o tempo de simulação aumenta.

Na próxima tabela são apresentados os resultados alcançados nos segundos testes de desempenho, mostrando igualmente os valores que foram utilizados em cada simulação, com exceção do tempo de simulação e do raio de cobertura dos APs que foram iguais para todos os testes, 1000s e 20m respetivamente.

Tabela 4.5 – Resultados dos segundos testes de desempenho do algoritmo para geração de *logs* RADIUS sintéticos

Área de Simulação (m^2)	Número de APs	Número de Stations	Tempo de execução (s)	Número de registos gerados
256	5	100	0,87	4342
484	10	200	1,58	9522
729	15	300	2,35	15250
1024	20	400	2,95	20532
1296	25	500	3,61	25440
1521	30	600	4,21	30862
1764	35	700	4,87	36096
2025	40	800	5,58	41890
2304	45	900	6,2	46476
2500	50	1000	6,95	52278

À semelhança dos primeiros testes, os resultados destes segundos testes, presentes na tabela anterior, foram igualmente ilustrados em gráfico. Neste segundo gráfico (figura 4.5) é igualmente representada a evolução do tempo de execução e do número de registos gerados mas neste caso à medida que aumenta a área de simulação utilizada no teste.

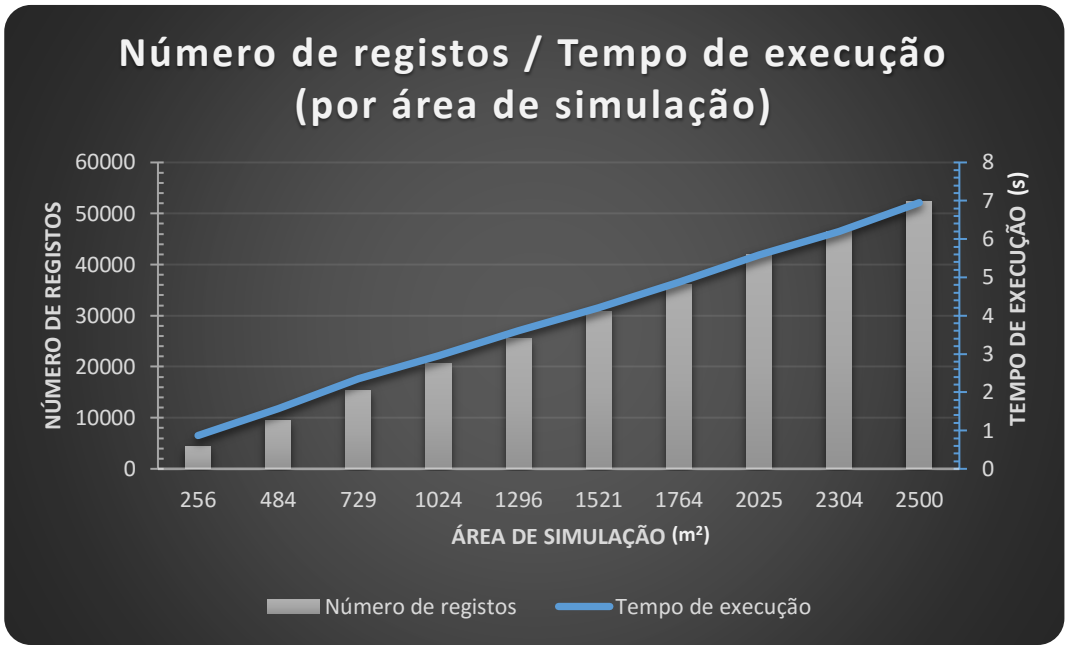


Figura 4.7 – Resultados dos segundos testes de desempenho do algoritmo para geração de *logs* RADIUS (por área de simulação).

Após representar graficamente os resultados dos segundos testes, torna-se ainda mais percetível a linearidade entre o número de registos gerados e a área de simulação, bem como para o tempo de execução. Como se pode verificar na figura 4.5, à medida que se aumenta a área de simulação (e os restantes parâmetros de forma proporcional), aumenta também, com uma linearidade bem definida, o tempo de execução do algoritmo e o número de registos gerados.

O tempo de execução do algoritmo, como foi explicado anteriormente, inclui não só gerar os registos mas também a sua escrita na base de dados. Nesse sentido, os tempos alcançados nos testes são perfeitamente aceitáveis, visto o número de registos que foram gerados e escritos na base de dados em cada período de tempo.

Assim, através destes dois conjuntos de testes de desempenho, é possível constatar que o algoritmo desenvolvido apresenta um desempenho positivo, indo ao encontro do que era pretendido. Verifica-se uma linearidade nos resultados, sob os dois pontos de vista apresentados anteriormente em forma de gráfico.

Em primeiro, mantendo o mesmo ambiente de simulação e variando apenas o tempo de simulação verificou-se um aumento linear do tempo de execução do algoritmo e do número de registos gerados à medida que se aumenta o tempo de simulação. Depois em segundo, alterando o ambiente de simulação de forma gradual e proporcional e mantendo o mesmo tempo de

simulação em todos os ambientes, o tempo de execução do algoritmo e o número de registos gerados aumentaram igualmente de forma linear.

Portanto, como era de esperar, a saída do algoritmo (registos) pode ser controlada através dos vários parâmetros de entrada. Nesse sentido, conclui-se então que em ambientes com características idênticas, quanto maior for o tempo de simulação maior será o número de registos gerados. De igual forma, se for mantido o mesmo tempo de simulação em ambientes com diferentes características, quanto maior for a área de simulação (e respetivos parâmetros de forma proporcional) maior será o número de registos gerados.

4.2 Algoritmo para extração de encontros

Este segundo algoritmo tem como objetivo extrair encontros de um *trace* de registos RADIUS. Este *trace* está presente numa base de dados e servirá de entrada para o algoritmo, sendo caracterizado no pseudocódigo deste algoritmo por traceRecords (Figura 4.8). Cada um dos registos de traceRecords tem o formato igual ao apresentado na descrição do algoritmo anterior, ou seja:

t, stationID, apID, eventType

Após a execução do algoritmo, o mesmo deverá gerar um conjunto de encontros para uma base de dados. Este conjunto de encontros é descrito no pseudocódigo do algoritmo como encountersList. Cada um dos encontros de encountersList é descrito através dos seguintes campos:

- **ApID:** identificação do AP ao qual as stations que formam o encontro estão associadas;
- **StationA_ID e StationB_ID**: identificação das stations que formam o encontro;
- **Start_Time:** instante de tempo em que o encontro se dá início ao encontro entre as duas stations no AP associado;
- **End_Time:** instante de tempo em que termina o encontro entre as duas stations no AP associado;

No final, os encontros gerados terão o seguinte formato:

ApID, StationA_ID, StationB_ID, Start-Time, End-Time

Para ser possível extrair os encontros através de um conjunto de registos RADIUS, é necessário inicialmente obter todos os APs únicos presentes no traceRecords, sendo guardados em uniqueAPList. Seguidamente, para cada um destes APs únicos (uniqueAP), obtém-se todos os registos associados ao AP, ou seja os que tenham o valor do campo ApID = uniqueAP. Estes registos são guardados numa lista denominada por recordsList.

Após este processo, a lista de registos é ordenada por ordem crescente do campo t para depois cada um dos registos ser sujeito a uma verificação. Esta verificação é feita pelo campo evenType, que descreve se o registo é do tipo "Start" ou "Stop". Se o registo for do tipo "Start", o mesmo é guardado num *buffer* denominado de bufferStartRecords. Caso seja um registo do tipo "Stop" é feita uma comparação entre este registo e cada um dos registos presentes em bufferStartRecords, nomeadamente no campo stationID. Sempre que os registos comparados não tenham o mesmo valor no campo stationID é gerado um encontro entre as duas stations dos dois registos comparados durante os tempos contidos no campo t de ambos. Caso contrário, é removido o registo de bufferStartRecords que está atualmente a ser comparado com o registo do tipo "Stop". Sempre que é gerado um encontro, o mesmo é guardado em encountersList e escrito na base de dados.

Seguidamente, é apresentada a descrição do algoritmo sob a forma de pseudocódigo.

Algoritmo para extração de encontros

traceRecords ← trace de logs *radius*;
uniqueAPList[k]: todos os APs únicos de traceRecords;
recordsList[n]: onde são guardados os registos que englobam o AP que está a ser processado;
recordsListSize: número de registos de recordsList;
bufferStartRecords[b]: onde são guardados os registos do tipo "Start";
bufferStartRecordsSize: número de registos de bufferStartRecords;

Entrada: *trace* de *logs* RADIUS, cada *log* do *trace* tem o formato: t, ApID, StationID, EventType;
Saída: encountersList, lista de encontros no formato: ApID, StationA_ID, StationB_ID, Start_Time, End_Time;

```
1. Início;
2. Ler traceRecords;
3. uniqueAPList[k] ← todos os Aps únicos de traceRecords;
4. Para cada AP contido em uniqueAPList:
5.     recordsList[r] ← traceRecords(uniqueAPList[uniqueAP]), todos os registos de traceRecords que
6.                      possuem o AP = uniqueAP;
7.     Ordenar registos de recordsList por ordem crescente de t;
8      Para cada i = 0 até recordsListSize – 1:
9.         Se recordsList[i].eventType == "START", então;
10.            bufferStartRecords[b] ← recordsList[i], adiciona registo i ao buffer;
11.            i ←   i + 1;
12.        Senão;
13.            Para j = 0 até bufferStartRecordsSize – 1:
14.               Se records[i].stationID ≠ bufferStartRecords[j].stationID  então:
15.                   encounter ← generateEncounter (unique_AP, recordsList[i].StationID,
16.                                bufferStartRecords[j].StationID, bufferStartRecords[j].time,
17.                                recordsList[i].time);
18.                encounterList ← encounter, adiciona encounter à lista de encontros;
19.                j ← j + 1;
20.            Fim Se;
21.        Fim Para;
22.        bufferStartRecords[j] ← removeRecord(), remove j do buffer;
23.    Fim Se;
24.  Fim Para;
25. Fim Para;
26.Fim.
```

Figura 4.8 – Pseudocódigo do algoritmo para extração de encontros

4.2.1 Implementação

À semelhança do que se fizera para o primeiro algoritmo, após o desenvolvimento do algoritmo para a extração de encontros, o próximo passo passava pela implementação do mesmo em JAVA.

Em termos de organização, seguiu-se o mesmo formato adotado no algoritmo anterior, ou seja, dividiu-se a implementação por *packages*, um que englobasse as classes referentes à base de dados, denominado *database* e outro com as classes referentes a todo a processo de extração de encontros, denominado *encounters*.

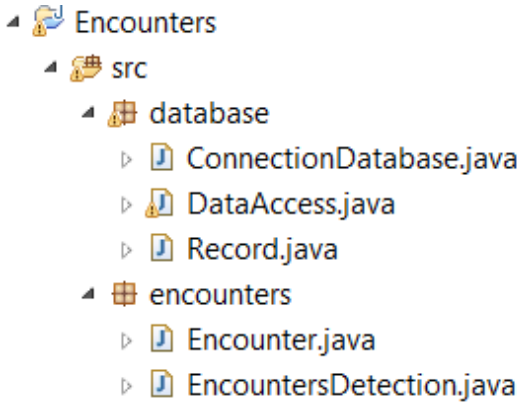


Figura 4.9 – Organização da implementação do algoritmo para extração de encontros.

No anexo 2 é apresentada toda a implementação deste algoritmo de forma mais detalhada.

4.2.2 Testes

À semelhança do que se fizera no algoritmo de geração de registos RADIUS sintéticos, neste segundo algoritmo fizeram-se igualmente testes de correção e de desempenho. O objetivo dos testes de correção passou pela verificação da correta geração dos encontros em termos de formato. Já nos testes de desempenho, o objetivo passou não só pela verificação dos tempos de execução do algoritmo para diferentes configurações de simulação mas também do número de encontros gerados face ao que foi descrito no algoritmo.

4.2.2.1 Testes de correção

A. Parâmetros considerados

Para os testes de correção do algoritmo para extração de encontros utilizou-se como entrada os registos gerados nos testes de correção do primeiro algoritmo. Assim, foram utilizados ao todo 3852 registos escritos anteriormente na base de dados através do algoritmo anterior, o de geração de *logs* RADIUS sintéticos. Em termos de análise, o parâmetro utilizado nestes testes de correção foi o formato do encontro gerado e a sua lógica.

Os registos a utilizar e previamente escritos na base de dados estão presentes na mesma sob o seguinte formato:

Figura 4.10 – Formato de cada registo na base de dados.

O campo id corresponde à identificação do registo, sendo esta identificação automaticamente incremental na base de dados.

A tabela seguinte resume todos os parâmetros considerados, os de entrada, saída e os de análise.

Tabela 4.6 – Parâmetros utilizados nos testes de correção do algoritmo para extração de encontros.

Tipo de parâmetro	Parâmetro	Valor
Entrada	Registos	3852
Saída	Encontros	————————
Análise	Formato e lógica do encontro	————————

B. Resultados e Análise

Dos 3852 registos sintéticos gerados anteriormente pelo primeiro algoritmo, foram extraídos ao todo 524694 encontros. A tabela seguinte apresenta esses encontros divididos por AP.

Tabela 4.7 – Resultados dos testes de correção do algoritmo para extração de encontros.

AP	Número de Encontros Extraídos
0	97
1	19349
2	90347
3	158758
4	256143
Total	**524694**

Os encontros extraídos e escritos na base de dados têm o seguinte formato:

Figura 4.11 – Formato de cada encontro na base de dados.

O campo id corresponde à identificação do encontro, sendo esta identificação semelhante à dos registos, ou seja, automaticamente incremental na base de dados.

De forma a analisar corretamente os dados, selecionou-se um conjunto de encontros e verificou-se manualmente se os mesmos estavam corretos, não só em relação ao seu formato mas também à lógica dos valores que cada campo continha. O conjunto de encontros selecionados estão associados ao AP0 e à Station66, podendo-se comprovar que os encontros foram corretamente formatados de acordo com o descrito no algoritmo.

id	ApID	StationA_ID	StationB_ID	Start_Time	End_Time
19	0	66	46	28	275
20	0	66	48	74	275
21	0	66	38	99	275
22	0	66	57	140	275
23	0	66	86	143	275
24	0	66	27	185	275
33	0	97	66	251	455
41	0	21	66	251	469
50	0	58	66	251	490
60	0	94	66	251	650
71	0	59	66	251	691
79	0	30	66	251	887
91	0	12	66	251	978

Figura 4.12 – Resultados dos testes de correção do algoritmo para extração de encontros: amostra de encontros associados ao AP0 e à Station66.

Para verificar a lógica dos encontros apresentados anteriormente, analisou-se individualmente cada registo de cada station envolvida nesses encontros, associados ao AP0.

id	t	StationID	ApID	EventType
2530	251	66	0	START
2531	275	66	0	STOP
1030	185	27	0	START
1031	206	27	0	STOP
1194	877	30	0	START
1195	887	30	0	STOP
1498	99	38	0	START
1499	137	38	0	STOP
2198	140	57	0	START
2199	148	57	0	STOP
2288	653	59	0	START
2289	691	59	0	STOP

2242	435	58	0	START
2243	490	58	0	STOP
3318	143	86	0	START
3319	197	86	0	STOP
3630	593	94	0	START
3631	650	94	0	STOP
3734	405	97	0	START
3735	455	97	0	STOP
1804	28	46	0	START
1805	43	46	0	STOP
514	935	12	0	START
515	978	12	0	STOP
826	452	21	0	START
827	469	21	0	STOP
1886	74	48	0	START
1887	113	48	0	STOP

Figura 4.13 – Conjunto de registos que deram origem aos encontros da figura 4.12.

De acordo com o especificado no algoritmo de extração de encontros, cada registo STOP "forma" um encontro com todos os registos START associados ao mesmo AP. Sendo assim e pelo facto de existirem um registo START e outro STOP para cada station, a Station66 "formará" no instante t = 275 (que será igual ao End_Time) um encontro com todas as outras stations no instante t dos seus registos START (que será igual Start_Time).

Por esta lógica, os encontros extraídos em que o End_Time fosse igual ao instante do registo STOP da Station66 (275) seriam os seguintes:

66, 66, 251, 275
66, 27, 185, 275
66, 30, 877, 275
66, 57, 140, 275
66, 59, 653, 275
66, 58, 435, 275
66, 86, 143, 275
66, 94, 593, 275
66, 97, 405, 275
66, 38, 99, 275
66, 46, 28, 275
66, 48, 74, 275
66, 12, 935, 275
66, 21, 452, 275

Mas também está especificado no algoritmo que só será extraído um encontro entre stations diferentes, ou seja, os registos STOP que têm associada uma determinada station não formam um encontro com um registo START associado à mesma station. Assim, ignora-se o primeiro encontro da lista anterior:

66, 66, 251, 275

Igualmente ao que sucedera com a station66, cada uma das outras stations também formam um encontro no seu instante t do registo STOP (que será igual ao End_Time) com todas as restantes stations nos seus instantes t dos registos START (que será igual ao Start_Time). Sendo assim, para além dos encontros anteriores seriam extraídos mais estes encontros onde está envolvida a station66:

27, 66, 251, 206
30, 66, 251, 887
38, 66, 251, 137
57, 66, 251, 148
59, 66, 251, 691
58, 66, 251, 490

86, 66, 251, 197
94, 66, 251, 650
97, 66, 251, 455
46, 66, 251, 43
48, 66, 251. 113
12, 66, 251, 978
21, 66, 251, 469

Sendo assim, somando todos os encontros no total seriam extraídos 26 encontros. Mas como se pode verificar, não faz sentido o instante de tempo de início do encontro (Start_Time) ser inferior ao instante de tempo final do encontro (End_Time), como acontece em alguns dos encontros apresentados anteriormente. Esses encontros são "ignorados" através da comparação do id das stations e se forem iguais é removido o registo do tipo START do buffer.

Assim, de acordo com esta especificação devem ser ignorados os seguintes encontros:

66, 30, 877, 275
66, 59, 653, 275
66, 58, 435, 275
66, 94, 593, 275
66, 97, 405, 275
66, 12, 935, 275
66, 21, 452, 275
27, 66, 251, 206
38, 66, 251, 137
57, 66, 251, 148
86, 66, 251, 197
46, 66, 251, 43
48, 66, 251. 113

Então, a lista dos encontros que devem ser extraídos neste teste é a seguinte:

66, 27, 185, 275
66, 57, 140, 275
66, 86, 143, 275
66, 38, 99, 275
66, 46, 28, 275
66, 48, 74, 275
30, 66, 251, 887
59, 66, 251, 691
58, 66, 251, 490
94, 66, 251, 650
97, 66, 251, 455
12, 66, 251, 978
21, 66, 251, 469

Ao comparar-se com a figura 4.12, onde são apresentados os encontros associados ao AP0 e à Station66 e que foram extraídos e escritos na base de dados, conclui-se que os encontros foram corretamente extraídos de acordo com o especificado no algoritmo.

4.2.2.2 Testes de desempenho

A. Parâmetros considerados

Após a implementação e a execução dos testes de correção estarem concluídos, o próximo passo passava por testar o desempenho do algoritmo.

Em termos de parâmetros são considerados dois tipos: entrada que corresponde aos registos sintéticos e de análise que engloba o tempo de execução do algoritmo e o número de encontros extraídos. Este tempo de execução corresponde não só à geração dos encontros mas também à escrita dos mesmos na base de dados.

De forma a testar da melhor forma o desempenho do algoritmo, definiu-se executar o mesmo 3 vezes para cada número de registos diferente, ao contrário do que se fizera no algoritmo anterior em que foram executadas 5 simulações para cada tempo de simulação. Foi definido desta forma pois entre os resultados do mesmo conjunto de registos apenas se pretendia verificar a linearidade dos valores do tempo de execução resultantes, uma vez que à partida o número de encontros extraídos em cada simulação com o mesmo conjunto de registos de entrada será o mesmo.

O formato dos registos que estão presentes na base de dados e que são utilizados nestes testes têm o formato apresentado anteriormente na figura 4.11.

A tabela seguinte resume os parâmetros utilizados nos testes de desempenho deste algoritmo.

Tabela 4.8 – Parâmetros utilizados nos testes de desempenho do algoritmo para extração de encontros.

Tipo de parâmetro	Parâmetro	Valor
Entrada	Registos	2000 - 10000
Análise	Tempo de execução	————————
	Número de encontros gerados	————————

B. Resultados e Análise

A tabela seguinte resume os resultados alcançados nos testes de desempenho deste segundo algoritmo. Na tabela é mostrado o número de registos que servem de entrada para este algoritmo, os tempos de execução alcançados em cada execução do mesmo e o número de encontros que foram extraídos. É igualmente apresentado a negrito o valor médio das 3 simulações de cada conjunto de registos, de forma a ser possível visualizar mais facilmente a linearidade dos resultados.

Tabela 4.9 – Resultados dos testes de desempenho do algoritmo de extração de encontros

Número de registos utilizados	Tempo de execução (s)	Número de encontros extraídos
2000	3,82	31.657
	3,73	31.657
	3,76	31.657
	3,77	
4000	16,04	130.542
	15,72	130.542
	16,61	130.542
	16,12	
6000	34,74	284.445
	35,98	284.445
	35,81	284.445
	35,51	
8000	62,33	505.414
	63,89	505.414
	64,48	505.414
	63,57	
10000	99,91	778.965
	97,52	778.965
	96,11	778.965
	97,85	

Como é lógico, o tempo de execução deste segundo algoritmo e o número de encontros que foram extraídos são dependentes do número de registos de entrada. Da mesma forma e como era esperado, do mesmo conjunto de registos foram extraídos o mesmo número de encontros nas 3 simulações.

De forma a analisar os resultados de forma mais detalhada, os mesmos foram representados em 3 gráficos diferentes. Um primeiro representando o tempo de execução do algoritmo em função do número de registos de entrada, outro com o número de encontros

extraídos em função do número de registos de entrada e por fim um com o tempo de execução em função do número de encontros extraídos. Seguidamente são apresentados esses mesmos gráficos.

Figura 4.14 – Resultados dos testes de desempenho do algoritmo para extração de encontros (Tempo de execução / Número de registos).

Como se pode observar no gráfico anterior, à medida que aumenta o número de registos de entrada aumenta também o tempo de execução do algoritmo. Apesar de ser visível uma pequena curva inicial, os resultados apresentam uma boa linearidade. Esta situação dá umas perspetivas iniciais positivas acerca do desempenho do algoritmo.

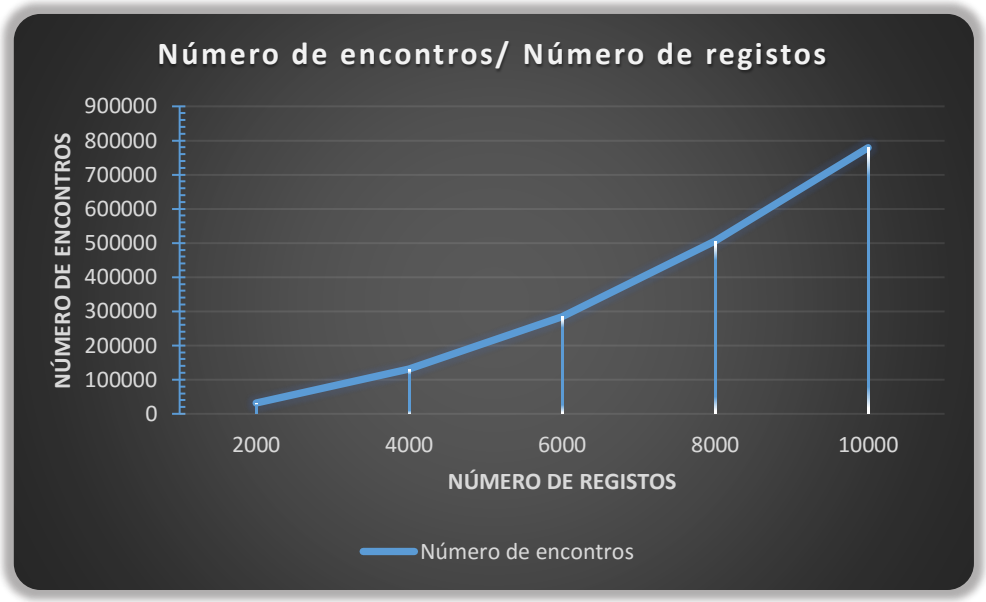


Figura 4.15 – Resultados dos testes de desempenho do algoritmo para extração de encontros (Número de encontros / Número de registos).

À semelhança do gráfico da figura 4.12, também este último referente ao número de encontros em função do número de registos apresenta uma boa linearidade. Como é possível constatar, à medida que se aumenta o número de registos de entrada, o tempo de execução de todo o algoritmo também aumenta de forma relativamente linear.

Figura 4.16 – Resultados dos testes de desempenho do algoritmo para extração de encontros (Tempo de execução / Número de encontros).

Como é possível verificar nesta terceira representação gráfica dos resultados, a linearidade mantêm-se nas mesmas proporções dos gráficos anteriores. O tempo de execução do algoritmo aumenta à medida que se aumenta o número de registos de entrada.

No final, após a visualização destes três gráficos, é possível constatar que o algoritmo apresenta um desempenho positivo, indo ao encontro do que era esperado. Era importante assegurar que o algoritmo apresentasse determinados comportamentos lógicos e básicos. Um dos mais lógicos é de que a partir do mesmo conjunto de registos fosse sempre extraído o mesmo número de encontros em qualquer simulação, situação que sucede como pôde ser visto anteriormente.

Para além disso, as outras situações evidenciadas são perfeitamente aceitáveis. É compreensível que quanto maior for o número de encontros extraídos maior é o tempo necessário para o algoritmo ser executado, uma vez que para além de extraí-los é necessário igualmente escrevê-los na base de dados. As restantes situações também são compreensíveis. É normal não só serem extraídos mais encontros a partir de um conjunto de registos maior, mas também que o tempo de execução aumente com o aumento do número de registos a processar.

4.3 Algoritmo para o encaminhamento epidémico de mensagens

Este terceiro algoritmo tem como objetivo a disseminação epidémica de mensagens numa rede oportunista, permitindo com isso extrair resultados com vista à avaliação do desempenho de protocolos de encaminhamento neste tipo de redes.

Para uma melhor compreensão dos pormenores inerentes a este algoritmo, pode-se descrever o mesmo dividindo-o em quatro partes: 1) processamento da lista de encontros; 2) geração de mensagens; 3) transmissão de mensagens; 4) extração de resultados.

1) Processamento da lista de encontros

Inicialmente, era importante definir em concreto qual a maneira mais adequada de processar toda a lista de encontros, considerando que este tipo de lista contém uma enorme quantidade de dados. Assim, optou-se por um processamento de encontros dividido por *slots* temporais, em que um *slot* corresponde à duração de tempo entre o primeiro instante de tempo da lista de encontros e o próximo instante de tempo. Como é lógico, à medida que se vai processando os encontros, os instantes são incrementados, permitindo processar todos os encontros de forma sequencial.

O primeiro passo da descrição deste algoritmo passou pela definição de quatro funções que permitissem obter os dados necessários para o processamento dos encontros.

A primeira função, denominada por *getUniqueStations*, possibilita obter todas as *stations* únicas existentes na lista de encontros que se pretende processar. No algoritmo, esta lista é guardada na variável uniqueStationsList.

A segunda função, *getEventsList()*, é responsável por obter uma lista com todos os tempos e respetivos de eventos associados (*Start/Stop*). Esta lista é obtida em ordem crescente dos tempos dos eventos e é guardada na variável eventsList.

A terceira função, denominada *updateRunningEncounters()* na descrição final de alto nível do algoritmo, permite definir quais os encontros que estão a decorrer em cada instante de tempo. Assim, para cada posição de eventsList definiu-se dois valores importantes: 1) eventTime, correspondente ao primeiro instante de tempo de eventsList; 2) eventTimeNext, correspondente ao próximo instante de eventsList. É através destes dois valores que se define cada *slot* temporal referido anteriormente, ou seja, cada um destes *slots* ocorre desde eventTime até ao eventTimeNext respetivo.

Após a definição destes dois instantes de tempo, processa-se a uma atualização dos encontros que estão a decorrer durante o atual *slot* temporal, encontros esses também denominados neste contexto de encontros ativos. Nesta atualização podem ocorrer dois cenários diferentes: 1) caso o eventTime corresponda ao início de um encontro, obtém-se os encontros que comecem em eventTime e adiciona-se os mesmos a runningEncounters; 2) caso o eventTime corresponda ao fim de um ou mais encontros, obtém-se os encontros que terminem em eventTime e remove-se os mesmos de runningEncounters. Após esta atualização, runningEncounters contém todos os encontros que estão ativos no atual *slot* temporal.

A quarta e última função desta primeira parte do terceiro algoritmo, *updateActiveStations()*, corresponde a uma atualização das *stations* envolvidas nos encontros a decorrer, denominadas neste contexto de *stations* ativas Logicamente, primeiro ocorre a atualização dos encontros e a partir desta atualização verifica-se quais as stations envolvidas, tornando esta função numa simples verificação das *stations* únicas existentes em runningEncounters. Em termos do algoritmo em si, estas *stations* ativas são guardadas na variável activeStations.

De salientar que durante cada um dos *slots* temporais existentes, apenas são processados os encontros que estão em curso durante esse *slot*, ou seja, entre o primeiro instante de tempo (eventTime) e o próximo correspondente (eventTimeNext).

Seguidamente é apresentada toda esta descrição do processamento dos encontros estruturada sob a forma de algoritmo de baixo nível.

Algoritmo de encaminhamento epidémico *1) Processamento da lista de encontros*

uniqueStationsList: lista de todas as stations representadas em todos os encontros (define o universo de destinatários possíveis)
eventsList: lista com todos os tempos (eventTime) e respetivos tipos de evento associados (início ou o fim de um encontro); tempos ordenados por ordem crescente
runningEncounters: lista de todos os encontros em curso (implica adicionar todos os encontros que começam em eventTime e remover todos os encontros que terminem em eventTime)
activeStations: lista de todas as stations ativas (todas as stations que estão envolvidas nos encontros em curso); define o universo de remetentes de mensagens
outputBuffer: fila de mensagens a transmitir de cada station ativa (activeStations)
outputBufferCapacity: capacidade das filas de mensagens de cada station
pmg: probabilidade de geração de mensagens
bandwidth: largura de banda
messageLength: tamanho de cada mensagem gerada
stationsNeighbors: lista de todas as stations que são vizinhas de cada station ativa (activeStations); corresponde às stations que formam encontro com a station ativa em runningEncounters
validMessageTime: corresponde ao tempo em que uma mensagem é considerada válida
transmitSlot: corresponde ao slot de transmissão de mensagens que está presente entre eventTime (instante de tempo atual de eventsList) e eventTimeNext (próximo instante de tempo de eventsList)
messageCounter ← 0, número de mensagens geradas;
deliveredMessageCounter ← 0, número de mensagens entregues;
transmitCounter ← 0, número de mensagens transmitidas;

Entrada: encountersList; validMessageTime; pmg; messageLength; bandwidth; OutputBufferCapacity; transmitSlot
Saída: instantes de tempo de geração e entrega de mensagens; número de mensagens geradas e entregues; cumprimento médio das filas de mensagens; número de mensagens transmitidas

```
1.  Início
2.  uniqueStationsList ← getUniqueStations(), obter lista de todas as stations únicas de
3.                                  encountersList;
4.  eventsList ← getEventsList(), obter lista de tempos ordenados e respetivos tipos de eventos
5.                  associados;
6.  Para cada event em eventsList:
7.      eventCounter ← 0;
8.      eventTime ← eventsList[eventCounter], primeiro instante de tempo de eventsList;
9.       eventTimeNext ← eventsList[eventCounter+1], próximo instante de tempo de eventsList;
10.      Enquanto eventTime <= eventTimeNext && eventCounter+1 < eventTimeListSize:
11.         Se eventTime corresponde ao início de um encontro, então:
12.            runningEncounters ← obter lista de encontros que comecem em eventTime e
13.                                  adicioná-los a runningEncounters;
14.         Senão (eventTime corresponde ao fim de um ou mais encontros);
15.            runningEncounters ← obter lista de encontros que terminem em eventTime e
16.                                  removê--los de runningEncounters;
17.         Fim Se;
18.         activeStations ← updateActiveStations();
```

Figura 4.17 – Pseudocódigo de baixo nível do algoritmo de encaminhamento epidémico referente ao processamento da lista de encontros.

2) Geração de mensagens

Esta parte do algoritmo corresponde à geração de mensagens por parte de cada *station* ativa. Na descrição de alto nível do algoritmo, a geração de mensagens é representada pela função *generateMessage()*. Nem sempre as *stations* geram mensagens, uma vez que a geração de mensagens é feita de acordo com o cálculo de um valor aleatório entre 0 e 1 (randomProbability) que é comparado com um valor probabilístico de geração de mensagens configurado inicialmente (pmg). Se o valor aleatório for menor que o valor probabilístico é gerada uma mensagem com origem na *station* ativa que está a ser processada para um destino aleatório entre todas as existentes na lista de encontros, ativas ou não, sendo o destino aleatório denominado no algoritmo de destinationStation. Caso contrário não é gerada qualquer mensagem com origem na *station* ativa que está a ser processada. Em cada *slot*, cada *station* ativa só gera, no máximo, uma mensagem com destino a uma outra *station*.

Uma mensagem é composta por um id (messageCounter), por um conteúdo (message), por uma origem (station), por um destino (destinationStation) e por um tempo de vida associado à mesma (ttl). Este ttl é representado pela soma do valor do eventTime atual com um valor previamente definido, representando o tempo em que cada mensagem é considerada válida. Como é lógico, não faz sentido gerar uma mensagem com a mesma origem e destino, por isso caso o destino aleatório seja igual à origem volta-se a selecionar aleatoriamente um novo destino até o destino ser diferente da origem. No final cada mensagem gerada é adicionada à fila de mensagens a transmitir da *station* que gerou (outputBuffer), ou seja a station ativa que está a ser processada no momento. Este processo de adição de mensagens à fila está ilustrado na figura 4.18.

Após gerar uma mensagem ainda é feita uma atualização dos vizinhos da station ativa que está a ser processada. As stations vizinhas são adicionadas a stationsNeighbors da station ativa correspondente, como é ilustrado na figura 4.18.

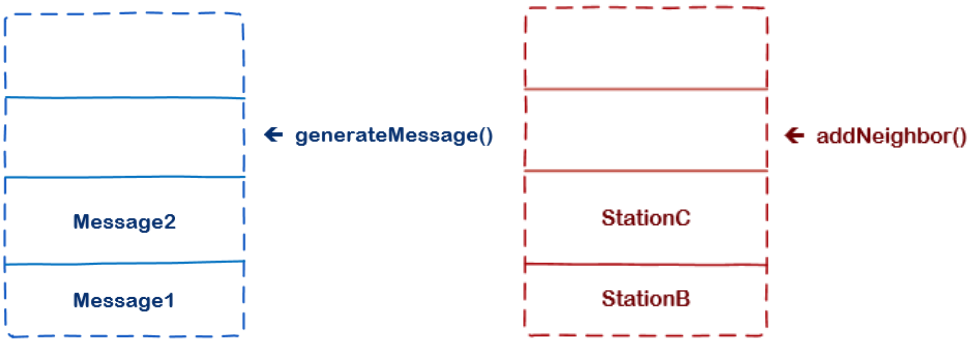


Figura 4.18 – Processo de adicionar mensagens geradas nas filas (esquerda) e de atualização da lista de vizinhos (direita).

No exemplo apresentado na figura anterior, a stationA gerou para a sua fila duas mensagens (Message1 e Message2) e terá que as transmitir para todos os vizinhos da sua lista (StationC e StationB).

Como acontecera na primeira parte, seguidamente é apresentada toda a descrição do processo de geração de mensagens estruturada sob a forma de algoritmo de baixo nível.

Algoritmo de encaminhamento epidémico *2) Geração de mensagens*

```
19.     Para cada station em activeStations:
20.         randomProbability ← número aleatório entre 0 e 1;
21.         Se randomProbabilityr < pmg, então:
22.             destinationStation ← random (uniqueStationsList), station aleatória de
23.                                  uniqueStationsList;
24.             Enquanto destinationStation == station:
25.                 destinationStation ← random (uniqueStationsList), station aleatória de
26.                                      uniqueStationsList;
27.             Fim Enquanto;
28.             ttl ← eventTime + validMessageTime;
29.             msg ← newMessage(messageCounter, message, station, destinationStation,
30.                       ttl);
31.             station.outputBuffer ← msg;
32.             messageCounter++;
33.         Fim Se;
34.         station.stationsNeighbors ← getStationNeighbors(), obter cada station vizinha e adicioná-
35.                                     a à lista de vizinhos da station ativa;
36.     Fim Para;
```

Figura 4.19 – Pseudocódigo de baixo nível do algoritmo de encaminhamento epidémico referente à geração de mensagens.

3) Transmissão de mensagens

Posteriormente à adição das mensagens geradas na fila de saída de cada station ativa correspondente, as mesmas têm que ser transmitidas durante os encontros. Nesta etapa são várias as considerações a ter em conta.

Em primeiro lugar deve-se considerar que as mensagens só podem ser transmitidas durante o *slot* temporal que está a ser processado, isto é, entre os dois instantes de tempo correspondentes (eventTime e eventTimeNext). Depois, dentro desse *slot* temporal, considera-se igualmente outro tipo de *slot* referente ao tempo que é gasto durante a transmissão de uma única mensagem, denominado no contexto de algoritmo por transmitSlot. Cada *slot* temporal (o primeiro referido) é composto por vários transmitSlot fixos e com a mesma duração temporal (valor definido inicialmente), como está ilustrado na figura 4.18. Em cada um destes transmitSlot é transmitida uma única mensagem a partir de uma única *station* ativa.

Sendo assim e como é lógico, pode acontecer que nem todas as mensagens sejam transmitidas, uma vez que, pode terminar o tempo disponível para a transmissão (primeiro *slot* temporal).

O processo de transmissão de uma mensagem é caracterizado pela adição da mensagem à fila da *station* que a irá receber (se a mensagem ainda não estiver contida na fila da referida station). Esse processo de transmissão é mostrado na figura 4.21.

Neste processo é importante destacar a gestão da fila de mensagens de cada station (outputBuffer) e da lista de vizinhos de cada uma dessas stations (stationsNeighbors), sendo as mesmas feitas de forma circular e como ilustrado na figura 4.22.

É selecionada a primeira station ativa que vai transmitir (nextStation) e a primeira mensagem que tem na sua fila. No primeiro transmitSlot é transmitida essa mensagem para o seu primeiro vizinho que está em stationNeighbors (targetStation), passando de seguida o apontador para o próximo vizinho e no final incrementa-se nextStation para dar a oportunidade de transmitir à próxima *station* ativa. Em cada uma destas stations ativas, a passagem para a próxima mensagem só é feita quando a atual tiver sido transmitida para todos os vizinhos contidos em stationsNeighbors, ou seja quando o apontador voltar apontar para o primeiro vizinho (firstNeighbor). Quando forem transmitidas todas as mensagens da fila de uma dada *station* ativa, a mesma é removida da lista de *stations* que transmitem mensagens.

Todo o processo de gestão e de transmissão de mensagens repete-se nos restantes transmitSlot disponíveis. Logicamente, em muitos casos pode acontecer que nem todas as

mensagens contidas nas filas sejam transmitidas, pois podem não existir transmitSlots suficientes no slot temporal principal que está a ser processado (eventTime – eventTimeNext).

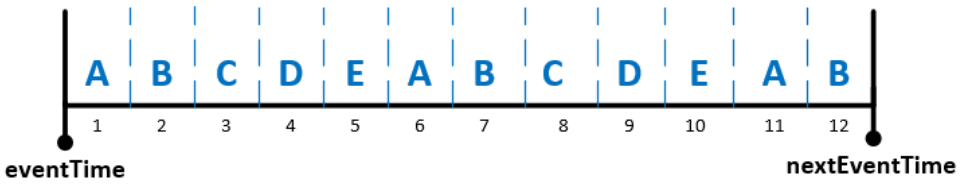


Figura 4.20 – Slot para a transmissão de mensagens dividido em transmitSlots (em cada transmitSlot uma station transmite uma única mensagem da sua fila para um dos seus vizinhos).

Seguindo o exemplo da figura 4.18 referente à fila de mensagens e à lista de vizinhos da StationA, neste *slot* temporal, a StationA transmitirá da seguinte forma:

1. Transmite a primeira mensagem (message1) para um dos vizinhos (StationB) no transmitSlot1;
2. Volta a transmitir a mesma mensagem para o outro vizinho (StationC) no transmitSlot6;
3. Finalmente transmite a mensagem seguinte (message2) para a StationB;
4. Como a StationA não dispõe de mais nenhum transmitSlot disponível, não transmite a segunda mensagem (message2) para o outro vizinho (StationC).

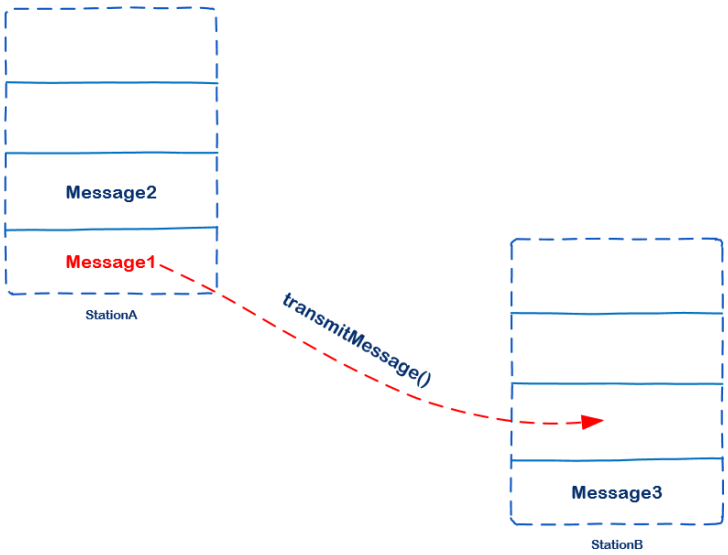


Figura 4.21 – Processo de transmissão de uma mensagem.

Como foi visto anteriormente, cada station possui uma fila de mensagens e uma lista de vizinhos associados. Para além disso, cada uma delas possui um "apontador" de forma a controlar qual a mensagem a ser transmitida e para que vizinho nos vários instantes de tempo.

No início da primeira transmissão de cada station, ambos os apontadores estão posicionados na primeira posição. Assim, será transmitida a mensagem contida na primeira posição da fila de mensagens da station respetiva para o vizinho posicionado na primeira posição da lista de vizinhos da mesma station, isto considerando que existe uma mensagem e um vizinho nessas posições. Esta situação pode ser verificada na figura 4.22 que ilustra o exemplo da stationA. Neste caso será transmitida a Message1 para a StationB.

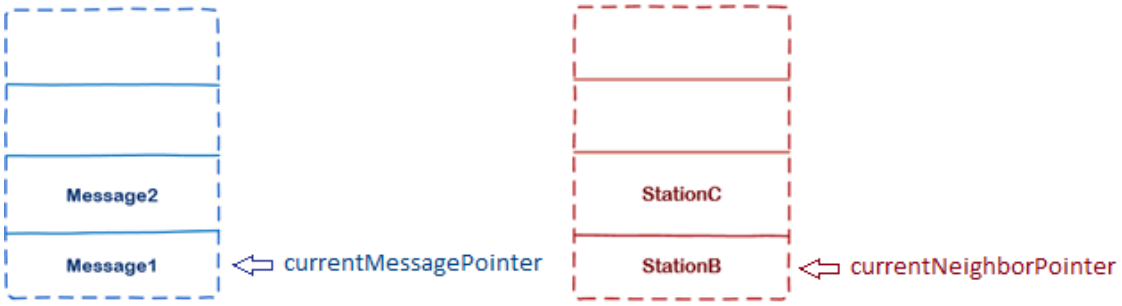


Figura 4.22 – Processo de funcionamento das filas de mensagens (esquerda) e das listas de vizinhos (direita) das stations ativas (I).

Após a transmissão, o apontador dos vizinhos avança uma posição e caso nessa próxima posição exista um vizinho, o apontador das mensagens mantém-se na primeira posição, uma vez que ainda existem mais vizinhos a quem transmitir a Message1.

De seguida, aquando da próxima oportunidade da stationA transmitir, o apontador das mensagens continua na primeira posição e o dos vizinhos aponta para a posição seguinte. Assim, na segunda oportunidade de transmissão, a stationA transmitirá a Message1 para a StationC. Esta situação está ilustrada na figura seguinte.

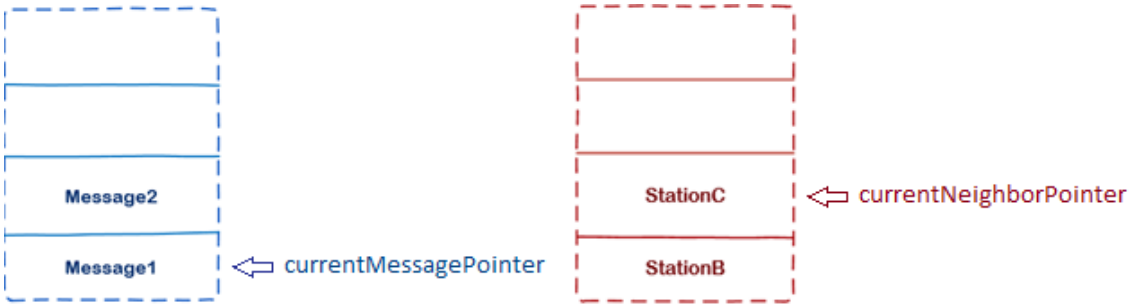


Figura 4.23 – Processo de funcionamento das filas de mensagens (esquerda) e das listas de vizinhos (direita) das stations ativas (II).

Seguidamente, o apontador dos vizinhos avança novamente uma posição mas como se pode verificar nas figuras anteriores, não existem mais vizinhos. Desta forma, o apontador dos vizinhos volta à primeira posição e o apontador das mensagens avança uma posição. Isto acontece pois a Message1 já foi enviada para todos os vizinhos (StationB e StationC).

Caso nessa próxima posição exista uma mensagem, a mesma terá que ser igualmente transmitida para todos os vizinhos existentes. Como existe, será transmitida a Message2 para a StationB. A figura seguinte ilustra a posição em que os apontadores estão nesta fase.

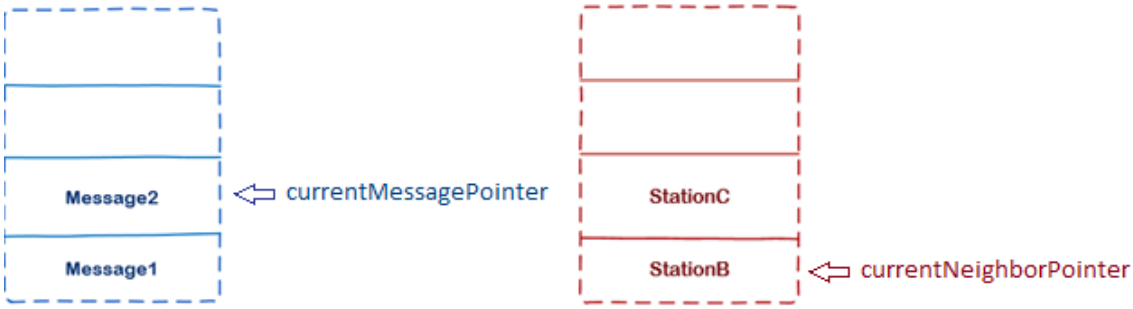


Figura 4.24 – Processo de funcionamento das filas de mensagens (esquerda) e das listas de vizinhos (direita) das stations ativas (III).

Na oportunidade de transmissão seguinte, a stationA irá transmitir a Message2 para o seu próximo vizinho (StationC), seguindo o mesmo processo anterior relativo à Message1. A figura 4.25 ilustra essa situação.

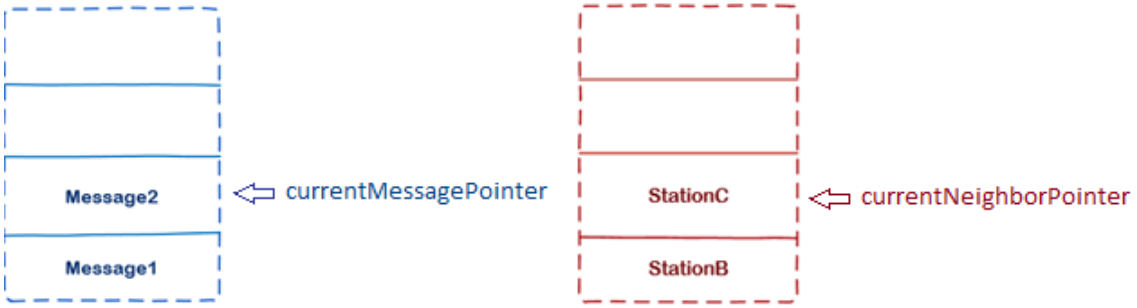


Figura 4.25 – Processo de funcionamento das filas de mensagens (esquerda) e das listas de vizinhos (direita) das stations ativas (IV).

Por fim, o apontador dos vizinhos volta para a primeira posição (não existem mais vizinhos atualmente) e como consequência disso, o apontador das mensagens avança igualmente. Como não existem mais mensagens a transmitir, o apontador das mensagens volta para a primeira posição.

Numa próxima situação de transmissão, a stationA irá transmitir novamente de forma ordenada. Caso seja transmitida uma mensagem para um vizinho que já a tenha na sua fila, não será adicionada uma cópia dessa mensagem à fila respetiva. Desta forma, é garantida a não existência de mensagens repetidas na fila da mesma station.

Seguidamente é apresentada toda a descrição do processo de transmissão de mensagens estruturada sob a forma de algoritmo de baixo nível.

```
37.        nextStation ← 0, primeira station em activeStations;
38.        Para cada transmitSlot de eventTime até eventTimeNext:
39.            targetStation ← getNextNeighbor();
40.            Se targetStation = = firstNeighbor, então:
41.                messagePointer ← getNextMessagePointer();
42.            Fim Se;
43.            Se messagePointer > nextStation.outputBuffer.size(), então:
44.                messagePointer ← firstMessagePointer;
45.                (remover nextStation da lista de stations que transmitem mensagens (activeStations);
46.            Senão;
47.                nextMessageToTransmit ← getMessage(messagePointer);
48.                targetStation.outputBuffer ← nextMessageToTransmit;
49.                transmitCounter++;
50.                Se targetStation == messagePointer.destinationStation, então:
51.                    deliveredMessageCounter++;
52.                Fim Se;
53.            Fim Se;
54.            nextStation ← getNextStation();
55.            Fim Para;
56.        Fim Enquanto;
57.    Fim Para;
58.    Fim.
```

Figura 4.26 – Pseudocódigo de baixo nível do algoritmo de encaminhamento epidémico referente à transmissão de mensagens.

4) Extração de resultados

Após toda a execução do algoritmo é essencial extraírem-se resultados. Para tal, era necessário colocar certas configurações ao longo do algoritmo.

Como foi dito na secção 3.2, referente à abordagem metodológica, as métricas que se pretendem extrair com este algoritmo são o atraso, a taxa de entrega de mensagens, o número de transmissões e o cumprimento médio das filas de mensagens.

Assim, de forma a ir ao encontro destas três métricas, ao longo do algoritmo é possível extraírem-se os tempos de entrega de cada mensagem que chega ao destino (após a transmissão da mensagem e a respetiva verificação se a mensagem chegou ao destino, e em caso afirmativo, é registado o instante de tempo em que a mesma chegou ao seu destino), o número de mensagens geradas (messageCounter) e o número de mensagens que são entregues (deliveredMessageCounter). Com isto, é possível extrair também várias variantes das referidas métricas, como o atraso mínimo, máximo, entre outros. Relativamente ao número de transmissões

(transmitCounter), sempre que o método referente à transmissão de mensagens é executado, esta variável é incrementada. É possível extrair outras variáveis desta métrica, como por exemplo a média de transmissões de mensagens por station. Por fim, a verificação do cumprimento médio das filas de mensagens de cada station é feito num determinado momento da simulação, recorrendo ao número de mensagens que estão guardadas nas filas de cada station.

Seguidamente é apresentada toda a abordagem explicada anteriormente numa versão de alto nível, ou seja mais resumida e simplificada.

Algoritmo de encaminhamento epidémico

uniqueStationsList: lista de todas as stations representadas em todos os encontros (define o universo de destinatários possíveis)
eventsList: lista com todos os tempos (eventTime) e respetivos tipos de evento associados (início ou o fim de um encontro); tempos ordenados por ordem crescente
runningEncounters: lista de todos os encontros em curso (implica adicionar todos os encontros que começam em eventTime e remover todos os encontros que terminem em eventTime)
activeStations: lista de todas as stations ativas (todas as stations que estão envolvidas nos encontros em curso); define o universo de remetentes de mensagens
outputBuffer: fila de mensagens a transmitir de cada station ativa (activeStations)
outputBufferCapacity: capacidade das filas de mensagens de cada station
pmg: probabilidade de geração de mensagens
bandwidth: largura de banda
messageLength: tamanho de cada mensagem gerada
stationsNeighbors: lista de todas as stations que são vizinhas de cada station ativa (activeStations); corresponde às stations que formam encontro com a station ativa em runningEncounters
validMessageTime: corresponde ao tempo em que uma mensagem é considerada válida
transmitSlot: corresponde ao slot de transmissão de mensagens que está presente entre eventTime (instante de tempo atual de eventsList) e eventTimeNext (próximo instante de tempo de eventsList)
messageCounter ← 0

Entrada: encountersList; validMessageTime; pmg; messageLength; bandwidth; OutputBufferCapacity; transmitSlot
Saída: instantes de tempo de geração e entrega de mensagens; número de mensagens geradas e entregues; cumprimento médio das filas de mensagens no final da simulação; número de mensagens transmitidas

```
1.  Início
2.  uniqueStationsList ← getUniqueStations();
3.  eventsList ← getEventsList();
4.  Para cada event em eventsList:
5.      runningEncounters ← updateRunningEncounters();
6.      activeStations ← updateActiveStations();
7.      Para cada station em activeStations:
8.          Se(message ← generateMessage(station) ≠ null), então:
9.              station.outputBuffer ← message;
10.             messageCounter++;
11.         Fim Se;
12.         stationNeighbors ← updateStationNeighbors(station);
13.     Fim Para;
14.     nextStation ← firstInActiveStations;
15.     Para cada transmitSlot de eventTime até eventTimeNext:
16.         nextStation.transmitMessage(transmitSlot);
17.         nextStation ← getNextStation();
18.     Fim Para;
19. Fim Para;
20. Fim.
```

Figura 4.27 – Pseudocódigo de alto nível do algoritmo de encaminhamento epidémico.

4.3.1 Implementação

Após o desenvolvimento do algoritmo deu-se início à fase de implementação, à semelhança do que sucedera com os anteriores algoritmos.

Relativamente à organização de toda a implementação deste algoritmo, seguiu-se novamente o formato adotado nos algoritmos anteriores, como pode ser visto na figura 4.24. Dividiu-se a implementação em *packages*, um que englobasse as classes referentes à base de dados, denominado *database*, outro com as classes referentes ao processo de disseminação epidémica de mensagens, denominado *epidemic*, outro com as classes dos objetos denominado *entities* e por fim um referente a todos os processos relacionados com as filas de saída das mensagens, denominado *queue*.

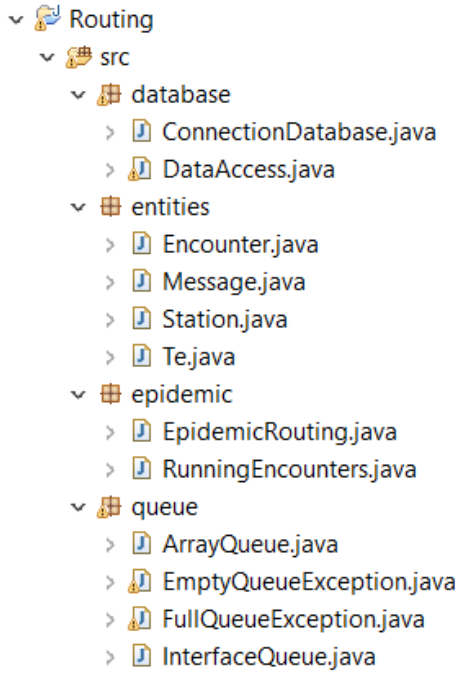


Figura 4.28 – Organização da implementação do algoritmo para o encaminhamento epidémico de mensagens.

De forma a otimizar o processo de transmissão de mensagens procedeu-se à especificação de uma outra estratégia de transmissão. Enquanto que na primeira estratégia explicada anteriormente só era possível transmitir uma mensagem em cada transmitSlot, nesta otimização cada station ativa tem oportunidade para transmitir uma mensagem em cada transmitSlot. Desta forma, tendo duas estratégias diferentes é possível extrair mais resultados e fazer um maior número de comparações e conclusões sobre os mesmos. Aquando da execução

deste terceiro algoritmo, será possível ao utilizador escolher qual das duas estratégias de transmissão é que pretende executar. Esta segunda estratégia está ilustrada na figura seguinte.

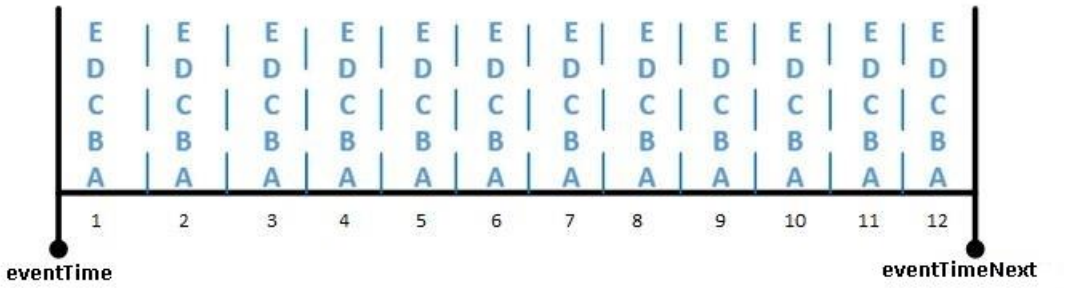


Figura 4.29 – Slot para a transmissão de mensagens na segunda abordagem (em cada transmitSlot cada station ativa pode transmitir uma mensagem da sua fila para um dos seus vizinhos).

No anexo 3 é apresentada toda a implementação deste algoritmo de forma mais detalhada, onde está incluída a especificação das duas diferentes estratégias de transmissão de mensagens.

4.3.2 Testes

À semelhança do que se fizera nos algoritmos anteriores, neste terceiro algoritmo fizeram-se igualmente testes de correção. O objetivo dos testes de correção passou pela verificação da correta extração dos resultados do encaminhamento epidémico de mensagens e da sua respetiva escrita na base de dados. Por outro lado não foram executados testes de desempenho neste algoritmo, uma vez que serão executadas experiências com o mesmo no próximo capítulo, permitindo com isso verificar o seu desempenho.

4.2.2.1 Testes de correção

A. Parâmetros considerados

Da mesma forma que foi feito nos testes de correção dos algoritmos anteriores, para este terceiro algoritmo foram igualmente considerados três diferentes tipos de parâmetros: de entrada, de saída e de análise. Todos estes parâmetros e os seus respetivos valores são apresentados seguidamente na tabela 4.10.

Tabela 4.10 – Parâmetros utilizados nos testes de correção do algoritmo para o encaminhamento epidémico de mensagens.

Tipo de parâmetro	Parâmetro	Valor
Entrada	Encontros	903
	Probabilidade de geração de uma mensagem	0.2
	Tempo de validade das mensagens	1500s
	Capacidade das filas de mensagens	20MB
	Taxa de transmissão	10Mbits
	Tamanho de cada mensagem	10KB
	Slot para a transmissão de mensagens	0.001s
Saída	Resultados do encaminhamento	————————
Análise	Escrita e formato dos resultados	————————

Os 903 encontros a utilizar nestes testes foram extraídos de um conjunto de 196 registos gerados através da configuração de simulação apresentada na tabela 4.11. Os encontros foram previamente escritos na base de dados e estão presentes na mesma sob o formato o seguinte formato:

id, ApID, StationA_ID, StationB_ID, Start_Time, End_Time

Este formato do encontro está de igual forma ilustrado anteriormente na figura 4.11

Tabela 4.11 – Parâmetros considerados para a geração dos registos nos testes de correção do algoritmo para o encaminhamento epidémico de mensagens

Tipo de parâmetro	Parâmetro	Valor
Simulação/Entrada	Área de simulação	$256m^2$
	Número de APs	5
	Raio de cobertura de cada AP	20
	Número de Stations	100
	Tempo de simulação	60s

No início dos testes é pedido que sejam introduzidos os valores pretendidos para os diferentes parâmetros de entrada. O parâmetro relativo ao slot para a transmissão de mensagens está dependente do tamanho da mensagem e da taxa de transmissão, por isso o valor temporal do mesmo é calculado automaticamente através da seguinte fórmula:

$$Slot = (\text{tamanho da mensagem} * 1024) / (\text{taxa de transmissão} \times 10^6)$$

Para além disso, é pedido igualmente que seja escolhida qual a estratégia de transmissão de mensagens que se pretende utilizar. Nestes testes serão testadas ambas, de forma a verificar se os resultados são corretamente extraídos e escritos na base de dados.

A figura seguinte ilustra o início dos testes utilizando a primeira estratégia de transmissão de mensagens.

```
Insira a probabilidade de geração de uma mensagem entre 0.0 e 1.0:
0,2
Insira o TTL das mensagens em segundos:
1500
Insira a taxa de transmissao em Mbps:
10
Insira o tamanho de cada mensagem em KB:
10
Insira a capacidade das filas de mensagens em MB:
20
-----------------------------------------------------------------------------------------------
Numero maximo de mensagens por fila: 2048
Valor do Slot de transmissão de mensagens: 0.001024s
-----------------------------------------------------------------------------------------------
Duas estrategias disponiveis para a transmissao de mensagens:

1)Em cada slot de transmissão apenas uma station ativa tem oportunidade de transmitir uma mensagem
2)Em cada slot de transmissão todas as stations ativas têm oportunidade de transmitir uma mensagem

Qual a estrategia que pretende utilizar? (1/2)
1
```

Figura 4.30 – Introdução dos valores dos parâmetros de entrada nos testes de correção do algoritmo para o encaminhamento epidémico de mensagens.

No final da execução do teste é possível visualizar os resultados do encaminhamento, como está ilustrado na figura seguinte.

```
##########################RESULTADOS##########################
------------------------------------------------------------
ATRASO:

Atraso medio: 0.2865152s
Atraso minimo: 0.002048s
Atraso maximo: 0.95744s
------------------------------------------------------------
ENTREGA DE MENSAGENS:

Numero de mensagens geradas:176
Numero de mensagens entregues:240
Taxa de entrega de mensagens:1.0 (136.36%)
Numero de mensagens unicas entregues:10
Taxa de entrega de mensagens unicas:0.06 (5.7%)
------------------------------------------------------------
COMPRIMENTO MEDIO DAS FILAS DE MENSAGENS NO FIM DA SIMULAÇAO:

Media de mensagens por fila:10.6
------------------------------------------------------------
NUMERO DE MENSAGENS TRANSMITIDAS:

Total de mensagens transmitidas:50834
Media de mensagens transmitidas por station:643.5
Numero de mensagens perdidas:50594
------------------------------------------------------------
RESULTADOS INSERIDOS NA BASE DE DADOS
--------------------------------------------
TEMPO DE EXECUCAO: 54239ms | 54s | 0min
```

Figura 4.31 – *Output* dos testes de correção do algoritmo para o encaminhamento epidémico de mensagens (primeira estratégia de transmissão de mensagens).

Como é possível visualizar na figura anterior os resultados extraídos apresentam um formato correto e de acordo com o expectável. Este processo foi repetido igualmente para a segunda estratégia de transmissão de mensagens. Essa situação pode ser verificada na figura

Os valores introduzidos nos parâmetros de entrada para a utilização da segunda estratégia de transmissão de mensagens foram idênticos aos apresentados na figura 4.29, com a exceção do último valor introduzido, que neste caso foi 2 em representação da segunda estratégia.

Na figura seguinte é possível constatar o output dos testes de correção para esta segunda estratégia e como se verifica foram igualmente extraídos no formato correto.

```
#########################RESULTADOS#########################
------------------------------------------------------------
ATRASO:

Atraso medio: 0.1087147s
Atraso minimo: 0.002048s
Atraso maximo: 0.447488s
------------------------------------------------------------
ENTREGA DE MENSAGENS:

Numero de mensagens geradas:168
Numero de mensagens entregues:11285
Taxa de entrega de mensagens:67.0 (6717.26%)
Numero de mensagens unicas entregues:12
Taxa de entrega de mensagens unicas:0.07 (7.1%)
------------------------------------------------------------
COMPRIMENTO MEDIO DAS FILAS DE MENSAGENS NO FIM DA SIMULAÇAO:

Media de mensagens por fila:13.8
------------------------------------------------------------
NUMERO DE MENSAGENS TRANSMITIDAS:

Total de mensagens transmitidas:685662
Media de mensagens transmitidas por station:8679.3
Numero de mensagens perdidas:674377
------------------------------------------------------------
RESULTADOS INSERIDOS NA BASE DE DADOS
---------------------------------------------
TEMPO DE EXECUCAO: 84577ms | 84s | 1min
```

Figura 4.32 – *Output* dos testes de correção do algoritmo para o encaminhamento epidémico de mensagens (segunda estratégia de transmissão de mensagens).

Para além do formato em que os resultados são extraídos, outro dos objetivos destes testes de correção é verificar o formato em que os mesmos são escritos na base de dados após a execução do algoritmo. Como ilustra a figura seguinte, os resultados de ambas as estratégias foram gravados corretamente na base dados.

transmission_strategy	average_delay	minimum_delay	maximum_delay	generated_msgs	delivered_msgs	delivered_rate	delivered_unique_msgs	unique_delivered_rate	transmited_msgs	average_transmited_msgs_station	lost_msgs	average_queue_lenght
2	0.1087147	0.002048	0.447488	168	11285	6717	12	7.14	685662	8679.27	674377	13.82
1	0.2865152	0.002048	0.95744	176	240	136	10	5.68	50834	643.47	50594	10.62

Figura 4.33 – Base de dados com os resultados dos testes de correção do algoritmo para o encaminhamento epidémico de mensagens.

CAPÍTULO 5
EXPERIÊNCIAS COM OS DADOS DISPONÍVEIS

Após o desenvolvimento dos algoritmos, a próxima etapa passava pela aplicação do algoritmo de encaminhamento epidémico num determinado *dataset*. Como foi dito anteriormente, esta fase era para ser executada utilizando os dados reais apresentados no capítulo 3 referente à abordagem metodológica. Contudo, por falta de tempo, não foi possível proceder dessa forma e optou-se pela utilização de dados sintéticos. Apesar disso, de forma aproximar os dados sintéticos aos dados reais, alterou-se o campo referente aos tempos dos encontros para o formato data (AAAA-MM-DD HH:MM:SS), ao invés de números inteiros, uma vez que esse é o formato usado nos dados reais.

Face a isto, o primeiro passo foi gerar um conjunto considerável de registos RADIUS sintéticos através do primeiro algoritmo desenvolvido. Seguidamente aplicou-se o segundo algoritmo de forma a extraírem-se encontros a partir dos registos gerados anteriormente. Por fim, aplicou-se o último algoritmo desenvolvido, referente ao encaminhamento epidémico de mensagens, foram encaminhadas mensagens e extraídos resultados através das métricas configuradas previamente.

Neste contexto, este capítulo visa descrever as experiências executadas em torno do algoritmo de encaminhamento epidémico de mensagens recorrendo ao referido *dataset*.

5.1 Dados

Antes de executarem-se as experiências com o algoritmo de encaminhamento era imprescindível a geração de um conjunto de registos para posteriormente extraírem-se os encontros a serem utilizados no encaminhamento. Dessa forma, utilizou-se o seguinte cenário de simulação:

Tabela 5.1 – Parâmetros considerados para a geração dos registos nas experiências.

Tipo de parâmetro	Parâmetro	Valor
Simulação/Entrada	Área de simulação	225m^2
	Número de APs	4
	Raio de cobertura de cada AP	20
	Número de Stations	80
	Tempo de simulação	140s

Após serem gerados os registos através deste cenário de simulação, extraíram-se um conjunto de encontros. Ao todo foram extraídos 3632 encontros e relembra-se que cada um deles possui na base de dados o formato ilustrado na figura 4.11, ou seja:

id, ApID, StationA_ID, StationB_ID, Start_Time, End_Time

A próxima figura mostra um desses encontros. O encontro ocorre entre a station66 e a station10, no ap0 e entre os instantes "2016-06-04 17:16:55" e "2016-06-04 17:17:00".

id ▲ 1	ApID	StationA_ID	StationB_ID	Start_Time	End_Time
1	0	66	10	2016-06-04 17:16:55	2016-06-04 17:17:00

Figura 5.1 – Exemplo de encontro presente na base de dados a utilizar nas experiências.

Em termos do espaço temporal, a lista de encontros a utilizar nas experiências regista um valor relativamente curto, nomeadamente um total de 8 minutos. Este tempo é contabilizado deste o início do primeiro encontro até ao início do último.

5.2 Experiências

Após a definição da lista de encontros a utilizar e antes de se dar início às experiências, o próximo passo passou pela escolha dos valores para os vários parâmetros de entrada do algoritmo de encaminhamento epidémico de mensagens. Seguidamente à escolha desses valores, iniciou-se a execução das diversas experiências. Todas estas escolhas e execuções são abordadas ao longo das próximas secções.

5.2.1 Parâmetros de entrada considerados

Para além da lista de encontros definida anteriormente, foram configurados mais alguns parâmetros de entrada no algoritmo de encaminhamento epidémico de mensagens. O primeiro

parâmetro configurado foi o "pmg" referente à probabilidade de geração de uma mensagem. Depois configurou-se o tempo em que uma mensagem mantém-se válida, refletida na variável "validMessageTime". Relativamente às filas de mensagens de cada station, configurou-se a variável "outputBufferCapacity" que permite configurar o limite de mensagens que cada station pode guardar. Quando este limite é atingido e uma nova mensagem tem que ser adicionada, é removida a mensagem que está na primeira posição da fila, dando esse lugar à nova mensagem. Por fim, foi configurada a taxa de transmissão, o tamanho de cada mensagem e por conseguinte o valor do transmitSlot (slot temporal destinado à transmissão de mensagens).

Como foi dito anteriormente, aquando do início da execução do algoritmo é pedido ao utilizador que introduza os valores que pretende para os parâmetros referidos anteriormente e que escolha qual a estratégia de transmissão pretendida para a simulação. Especificamente para estas experiências, os valores utilizados são os apresentados na tabela 5.2.

Tabela 5.2 – Parâmetros de entrada considerados para o encaminhamento epidémico de mensagens nas experiências realizadas.

Tipo de parâmetro	Parâmetro	Valor
Simulação/Entrada	Probabilidade de geração de uma mensagem	0.2
	Tempo de validade das mensagens	3600s
	Capacidade das filas de mensagens	10MB
	Taxa de transmissão	10Mbits
	Tamanho de cada mensagem	4 – 20KB
	Slot para a transmissão de mensagens	0.0005 – 0.01s

5.2.2 Execução das experiências

Inicialmente foram executadas 10 experiências com os valores especificados anteriormente, 5 para cada estratégia de transmissão de mensagens. Todas estas experiências foram feitas utilizando quatro valores fixos, variando apenas em dois valores, nomeadamente o tamanho de cada mensagem e por consequência o slot para a transmissão de mensagens (que

está dependente do tamanho da mensagem e da taxa de transmissão). Assim, foram executadas as seguintes experiências:

➔ **Primeira estratégia de transmissão de mensagens:**

Experiência 1: Mensagem: 4KB | Slot para a transmissão de mensagens: 0.0004s
Máximo de mensagens na fila de cada station: 2560 (10MB)

Experiência 2: Mensagem: 8KB | Slot para a transmissão de mensagens: 0.0008s
Máximo de mensagens na fila de cada station: 1280 (10MB)

Experiência 3: Mensagem: 12KB | Slot para a transmissão de mensagens: 0.0012s
Máximo de mensagens na fila de cada station: 853 (10MB)

Experiência 4: Mensagem: 16KB | Slot para a transmissão de mensagens: 0.0016s
Máximo de mensagens na fila de cada station: 640 (10MB)

Experiência 5: Mensagem: 20KB | Slot para a transmissão de mensagens: 0.002s
Máximo de mensagens na fila de cada station: 512 (10MB)

➔ **Segunda estratégia de transmissão de mensagens:**

Experiência 6: Mensagem: 4KB | Slot para a transmissão de mensagens: 0.0004s
Máximo de mensagens na fila de cada station: 2560 (10MB)

Experiência 7: Mensagem: 8KB | Slot para a transmissão de mensagens: 0.0008s
Máximo de mensagens na fila de cada station: 1280 (10MB)

Experiência 8: Mensagem: 12KB | Slot para a transmissão de mensagens: 0.0012s
Máximo de mensagens na fila de cada station: 853 (10MB)

Experiência 9: Mensagem: 16KB | Slot para a transmissão de mensagens: 0.0016s
Máximo de mensagens na fila de cada station: 640 (10MB)

Experiência 10: Mensagem: 20KB | Slot para a transmissão de mensagens: 0.002s
Máximo de mensagens na fila de cada station: 512 (10MB)

Seguidamente, no próximo capítulo, são apresentados todos os resultados destas experiências e a respetiva análise detalhada aos mesmos.

CAPÍTULO 6
RESULTADOS E DISCUSSÃO

Ao longo deste capítulo são apresentados os resultados alcançados em todas as experiências executadas, sob a forma de tabelas. Após isso é feita a respetiva análise aos mesmos, fazendo nesse sentido diversas comparações de resultados através de gráficos, nomeadamente entre as duas diferentes estratégias de transmissão de mensagens utilizadas.

6.1 Resultados alcançados

Como foi referido anteriormente, as principais métricas que se pretendiam extrair eram essencialmente quatro, mais especificamente o atraso, a taxa de entrega de mensagens, o cumprimento médio de mensagens nas filas e o número de mensagens transmitidas por station. A partir destas quatro métricas principais, alargou-se as mesmas às suas diversas variantes. No caso do atraso, extraiu-se o atraso médio, mínimo e máximo. Relativamente à taxa de entrega de mensagens, extraiu-se não só a taxa de entrega de mensagens únicas mas também de todas as mensagens entregues, sejam únicas ou não. Por fim, extraiu-se o total de mensagens transmitidas na simulação, a média do total de mensagens transmitidas por station e o número de mensagens perdidas (foram transmitidas mas acabaram por não serem entregues).

6.1.1 Primeira estratégia de transmissão de mensagens

A tabela seguinte apresenta todos os resultados alcançados nas 5 experiências que foram feitas utilizando a primeira estratégia de transmissão de mensagens. Os resultados estão divididos pelos 5 tamanhos de mensagem diferentes utilizados, refletindo-se cada tamanho numa experiência distinta. Os valores apresentados na tabela, referentes ao número de mensagens transmitidas por station, são valores médios arredondados. Da mesma forma, os valores apresentados do cumprimento médio das filas também são valores arredondados e referem-se ao número de mensagens presentes nas filas das várias stations no fim da execução de cada experiência.

Tabela 6.1 – Resultados das experiências através da primeira estratégia de transmissão de mensagens.

	Tamanho da mensagem				
Métrica	**4KB**	**8KB**	**12KB**	**16KB**	**20KB**
Atraso mínimo	0.005s	0.084s	0.511s	0.061s	0.131s
Atraso médio	**11.209s**	**2.563s**	**7.675s**	**9.383s**	**10.674s**
Atraso máximo	72.021s	25.018s	21.224s	65.252s	25.323s
Mensagens geradas	554	581	542	581	543
Mensagens entregues	9540	9958	5747	2905	2823
Mensagens únicas entregues	9	11	6	12	9
Taxa de entrega	1722.02%	1713.94%	1060.33%	500.00%	519.89%
Taxa de entrega única	**1.62%**	**1.89%**	**1.11%**	**2.07%**	**1.66%**
Mensagens transmitidas	1044962	518317	345423	261294	207771
Média mensagens transmitidas por station	**13062**	**6479**	**4318**	**3266**	**2597**
Mensagens perdidas	1035422	508359	339676	258389	204948
Cumprimento médio das filas	**15**	**19**	**13**	**19**	**16**
Tempo de execução	57min	27min	19min	15min	11min

Para além desta tabela principal, seguidamente é apresentada uma outra tabela com o número de stations que transmitem um determinado número de mensagens.

Tabela 6.2 – Outros resultados: número de stations que transmitem um determinado número de mensagens (primeira estratégia de transmissão de mensagens).

Número de Transmissões	4KB	8KB	12KB	16KB	20KB
0 – 2500	0	4	26	45	79
2500 – 5000	2	25	36	35	1
5000 – 10000	12	23	12	0	0
10000– 15000	19	17	6	0	0
> 15000	47	11	0	0	0

6.1.2 Segunda estratégia de transmissão de mensagens

Da mesma forma que se fez para a primeira estratégia de transmissão de mensagens, é apresentada seguidamente uma tabela com os resultados alcançados nas 5 experiências, desta vez realizadas com a segunda estratégia. A disposição da tabela e o tipo de valores contidos na mesma são exatamente idênticos à tabela dos resultados da primeira estratégia.

Tabela 6.3 – Resultados das experiências através da segunda estratégia de transmissão de mensagens.

	Tamanho da mensagem				
Métrica	**4KB**	**8KB**	**12KB**	**16KB**	**20KB**
Atraso mínimo	0.001s	0.002s	0.015s	0.005s	0.004s
Atraso médio	**66.007s**	**41.108s**	**27.135s**	**40.348s**	**27.773s**
Atraso máximo	288.001s	388.001s	212.029s	324.002s	220.152s
Mensagens geradas	547	543	552	537	551
Mensagens entregues	182476	122085	84199	86283	56773
Mensagens únicas entregues	12	11	9	13	14
Taxa de entrega	33359.41%	22483.43%	15253.44%	16067.60%	10303.63%
Taxa de entrega única	**2.19%**	**2.03%**	**1.63%**	**2.42%**	**2.54%**
Mensagens transmitidas	18199006	9093504	6065443	4553021	3643051
Mensagens transmitidas por station	**227488**	**113669**	**75818**	**56913**	**36431**
Mensagens perdidas	18016530	8971419	5981244	4466738	3586278
Cumprimento médio das filas	**18**	**18**	**17**	**18**	**22**
Tempo de execução	55min	27min	18min	14min	11min

É apresentada igualmente uma nova tabela com o número de stations que transmitem um determinado número de mensagens, referente a esta segunda estratégia de transmissão de mensagens. Dada à maior quantidade de mensagens transmitidas nesta estratégia, a escala do número de transmissões na tabela é igualmente superior.

Tabela 6.4 – Outros resultados: número de stations que transmitem um determinado número de mensagens (segunda estratégia de transmissão de mensagens).

Número de Transmissões	4KB	8KB	12KB	16KB	20KB
0 – 50000	1	6	16	46	51
50000 – 100000	4	30	46	34	29
100000 – 150000	8	27	18	0	0
150000 – 200000	20	16	0	0	0
> 200000	47	1	0	0	0

6.2 Análise e discussão de resultados

Após alcançar os resultados apresentados anteriormente, era imprescindível proceder à sua análise. Essa análise focou-se essencialmente na comparação entre as duas estratégias de transmissão de mensagens. Para uma melhor visualização das análises, recorreu-se à criação de diversos gráficos.

Nesse sentido, o primeiro gráfico a ser criado tinha como objetivo comparar os valores do atraso médio que foram alcançados nas duas estratégias de transmissão de mensagens, utilizando mensagens de diferentes tamanhos. A figura 6.1 ilustra esse mesmo gráfico.

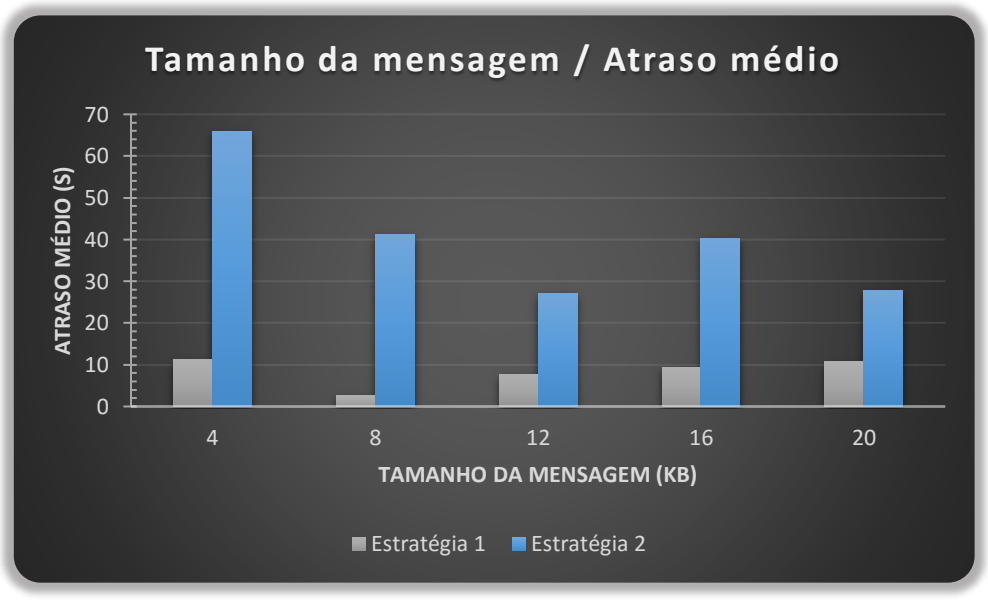


Figura 6.1 – Análise aos resultados alcançados com as duas diferentes estratégias de transmissão (Tamanho da mensagem / Atraso médio).

Como é possível constatar no gráfico anterior, os atrasos médios alcançados nas duas estratégias apresentam uma discrepância bem visível. Em todas as simulações executadas, os valores médios de atraso alcançados na segunda estratégia foram sempre superiores aos da primeira, onde essa maior superioridade verificou-se quando foram utilizadas mensagens de 4KB, em que a diferença no atraso foi de cerca de 55s. Face a isto, é possível constatar que o atraso médio possui uma dependência direta em relação à estratégia de transmissão de mensagens. Por outro lado, é igualmente possível constatar que não existe dependência direta em relação ao tamanho da mensagem. Este facto resulta essencialmente da não existência de uma tendência visível nos resultados entre os diferentes tamanhos de mensagem.

Apesar das DTNs serem caracterizadas pelos seus atrasos longos e variáveis e de serem tolerantes aos mesmos, não é de desprezar que se tente alcançar valores baixos.

Seguidamente ao atraso médio, era importante fazer uma comparação direta entre a taxa de entrega única alcançada nas duas estratégias de transmissão de mensagens. Dessa forma, o próximo gráfico representa essa mesma comparação.

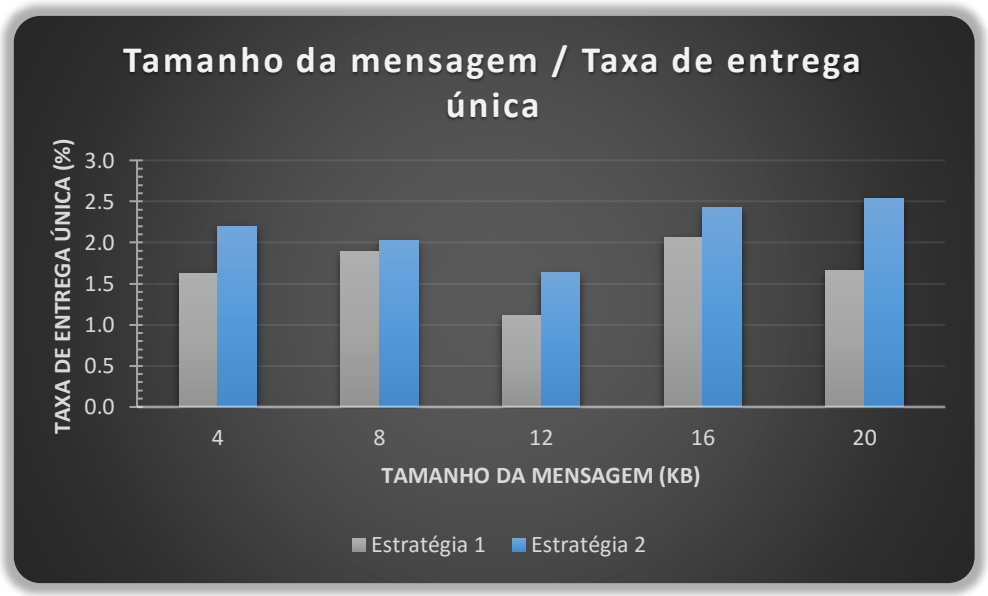


Figura 6.2 – Análise aos resultados alcançados com as duas diferentes estratégias de transmissão (Tamanho da mensagem / Taxa de entrega única).

Como se pode visualizar no gráfico da figura 6.2, a taxa de entrega única da estratégia 2 é sempre superior à taxa da estratégia 1. A taxa apresenta o seu máximo (cerca de 2.5%) utilizando a estratégia 2 e um tamanho de mensagem de 20KB e o seu mínimo (cerca de 1.1%) utilizando a estratégia 1 e quando foram utilizadas mensagens de 12KB.

À medida que se vai aumentando o tamanho da mensagem, a taxa da estratégia 2 apresenta uma descida relativamente linear, excetuando quando se utilizam mensagens de 16 e 20KB, pois os valores voltam a subir. Enquanto que na primeira estratégia, na transição das mensagens de 16KB para as de 20KB a taxa desceu ligeiramente, na segunda estratégia aconteceu o inverso, ou seja uma pequena subida. Esta pequena subida fez com que se alcançasse o máximo já referido.

Em termos de valores das taxas de entrega única alcançados em ambas as estratégias, apesar de serem baixos podem ser também considerados aceitáveis, pois as DTNs são redes caracterizadas por terem baixas taxas de entrega. Apesar das taxas alcançadas serem baixas, há que considerar que todas as experiências foram executadas recorrendo a um conjunto fixo relativamente curto de encontros e número de stations associadas, facto que pode acentuar a baixa taxa de entrega. Da mesma forma, o facto de cada station ativa poder gerar uma mensagem em cada evento faz com que possa existir uma grande quantidade de mensagens únicas na rede e que não existam encontros e tempo suficientes para fazê-las chegar ao destino. O reduzido tempo disponível para efetuar o encaminhamento das mensagens torna-o na principal causa das baixas taxas de entrega única verificadas.

Apesar da esperada superioridade que se verificou na segunda estratégia, a amplitude dessa superioridade não é assim tão grande como igualmente se poderia esperar (dada à maior quantidade de transmissões efetuadas com a segunda estratégia). A verdade é que apesar de serem feitas mais transmissões com a segunda estratégia, há que ter em consideração que em muitos casos a mensagem a transmitir é uma cópia de uma mensagem que já existe na fila da station alvo e dessa forma não é adicionada à referida fila. Mesmo assim, é natural que sejam adicionadas mais mensagens nas filas quando se utiliza a segunda estratégia e por isso, algumas dessas filas podem atingir o seu limite e como tal algumas mensagens são removidas das mesmas, reduzindo a taxa de entrega única (caso essas mensagens removidas ainda não tenham sido entregues uma única vez).

Face a tudo isto, é igualmente esperado que se atinjam taxas de entrega (não únicas) superiores na estratégia 2 em relação à estratégia 1, uma vez que são feitas mais transmissões

em cada transmitSlot, aumentando a possibilidade de existência na rede de um maior número de cópias da mesma mensagem. Essa situação acontece, como se pode verificar no gráfico seguinte.

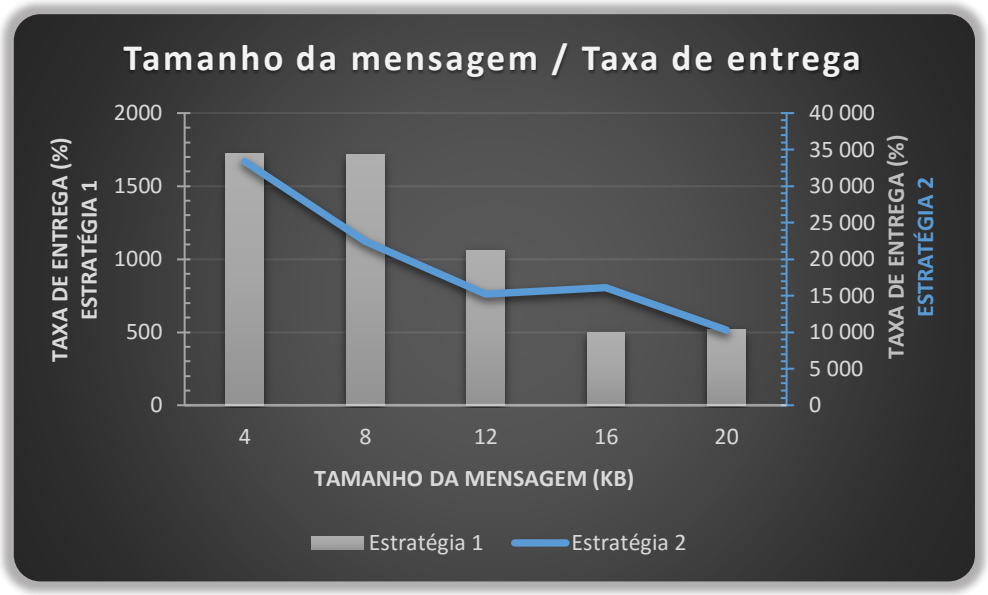


Figura 6.3 – Análise aos resultados alcançados com as duas diferentes estratégias de transmissão (Tamanho da mensagem / Taxa de entrega não única).

Como se pode visualizar no gráfico da figura 6.3, a taxa de entrega de mensagens não únicas é sempre superior na estratégia 2. Há que ter em atenção que os valores da taxa referente à estratégia 2 são os do eixo vertical do lado direito (0% – 40000%), enquanto o da estratégia 1 é o vertical do lado esquerdo (0% – 2000%).

Seguidamente apresenta-se a análise gráfica relativa ao cumprimento médio das filas de mensagens das stations no final da execução do algoritmo de encaminhamento.

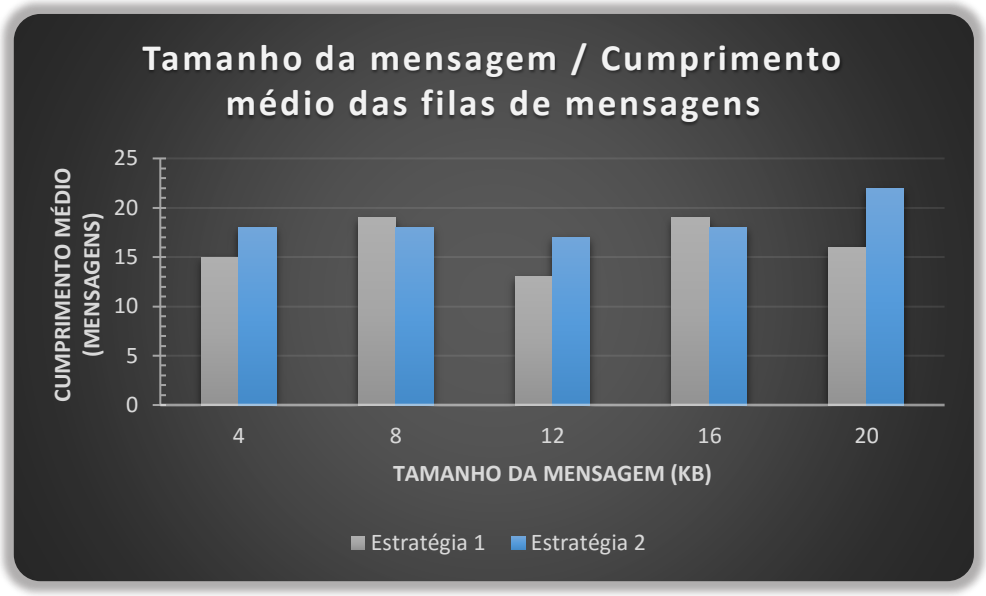


Figura 6.4 – Análise aos resultados alcançados com as duas diferentes estratégias de transmissão (Tamanho da mensagem / Cumprimento médio das filas de mensagens).

Como se pode verificar na figura anterior, a média de mensagens existentes nas filas das stations no final das várias simulações é relativamente próximo entre as duas estratégias. A segunda estratégia é superior à primeira em mais uma ocasião que esta, nomeadamente nas simulações utilizando mensagens de 4, 12 e 20KB. Logicamente, a primeira estratégia apresenta valores superiores nas restantes simulações, ou seja quando foram utilizadas mensagens de 8 e 16KB. Outra situação evidenciada é o facto de na segunda estratégia os cumprimentos médios das filas no fim das 5 simulações serem sempre muito semelhantes, verificando-se a presença de cerca de 17/18 mensagens nas filas de cada simulação. A única exceção ocorre quando foram utilizadas mensagens de 20KB onde o cumprimento médio verificado atingiu o seu máximo de todas as simulações, com cerca de 22 mensagens.

Em termos da relativa proximidade verificada entre os cumprimentos médios alcançados nas diferentes experiências, esta situação pode ter resultado do facto do limite das filas ser fixo (10MB).

Relativamente às transmissões de mensagens em cada station, analisa-se graficamente de seguida na figura 6.4. Nesse gráfico, o eixo referente aos valores da primeira estratégia é o vertical do lado esquerda, enquanto que o vertical do lado direito é referente à segunda estratégia.

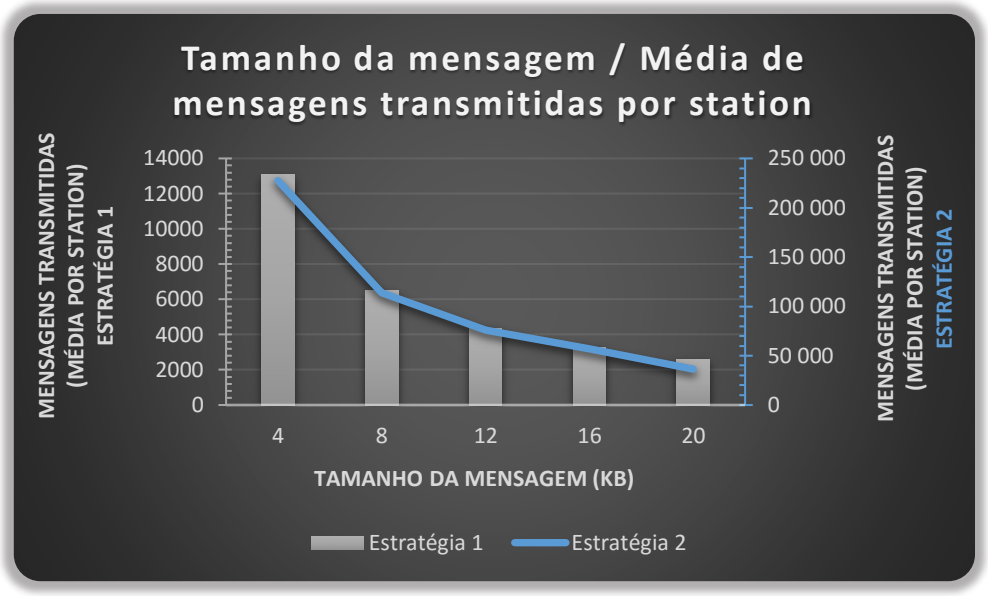


Figura 6.5 – Análise aos resultados alcançados com as duas diferentes estratégias de transmissão (Tamanho da mensagem / Média de mensagens transmitidas por station).

Como se pode visualizar no gráfico anterior, a média de mensagens transmitidas por station na segunda estratégia é sempre superior à primeira estratégia, o que é perfeitamente normal, dada ao maior número de transmissões que são feitas em cada transmitSlot ao utilizar-se a segunda estratégia. Enquanto que na primeira estratégia uma única station tem a possibilidade de transmitir uma única mensagem em cada transmitSlot, na segunda, todas as stations ativas têm possibilidade transmitir uma mensagem em cada um desses slots.

Outro facto que pode ser visível no gráfico anterior, é que o número médio de transmissões nas duas estratégias vai reduzindo à medida que se aumenta o tamanho das mensagens. Este facto é completamente normal, uma vez que quanto maior for a mensagem maior será o valor de um transmitSlot e como consequência disso menos mensagens poderão ser transmitidas entre o eventTime e o eventTimeNext (o tempo entre estas duas variáveis é divido em transmitSlots, logo quanto maior for um transmitSlot menos transmitSlots existirão nesse tempo).

Dado que as stations são nós móveis e que por norma possuem baterias relativamente limitadas, estes resultados preveem que as stations da segunda estratégia fiquem sem energia mais rapidamente, dada à enorme diferença na quantidade de transmissões que efetuam em comparação com a primeira estratégia. Esta situação pode fazer com que algumas mensagens nunca sejam entregues, uma vez que as stations que as possuem podem ficar sem energia após um determinado número de transmissões. Este facto levará a uma redução do número de mensagens entregues.

Continuando ainda no número de mensagens transmitidas por station, analisam-se as tabelas 6.3 e 6.4, referentes ao número de stations que transmitem um determinado número de mensagens em cada uma das estratégias respetivamente. Relativamente à primeira estratégia (tabela 6.3), à medida que se aumenta o tamanho da mensagem, o número de stations que transmitem até 2500 vezes começa a aumentar gradualmente e em contrapartida o número de stations que transmitem acima desse valor começa a reduzir gradualmente a partir dos 12KB. A partir dos mesmos 12KB já nenhuma station transmite acima de 15000 vezes, sendo que seguidamente nos 16KB já nenhuma transmite acima de 5000 vezes (45 transmitem entre 0 e 2500 vezes e 35 entre 2500 e 5000 vezes). No final quando se utilizam mensagens de 20KB praticamente todas as stations transmitem até 2500 vezes, onde apenas uma única station transmite entre 2500 e 5000 vezes.

Já utilizando a segunda estratégia (tabela 6.4), onde foi utilizada uma escala diferente dada à maior quantidade de transmissões (0 a >200000), de inicio (4KB) a maior parte das stations acentua-se no maior valor da escala das transmissões (> 200000). Nestes 4KB de tamanho de mensagem apenas uma station transmite o menor número de vezes da escala, ou seja entre 0 a 50000 vezes. À medida que se aumenta o tamanho da mensagem, naturalmente as stations vão transmitindo menos vezes, tendendo aproximar-se dos valores da escala de transmissões mais baixos. No final, com 20KB de tamanho de mensagem, todas as stations transmitem até 100000 vezes.

Seguidamente é analisado o tempo de execução decorrido nas diversas experiências executadas. A figura que se segue representa uma comparação direta entre os tempos de execução alcançados nas duas estratégias de transmissão de mensagens.

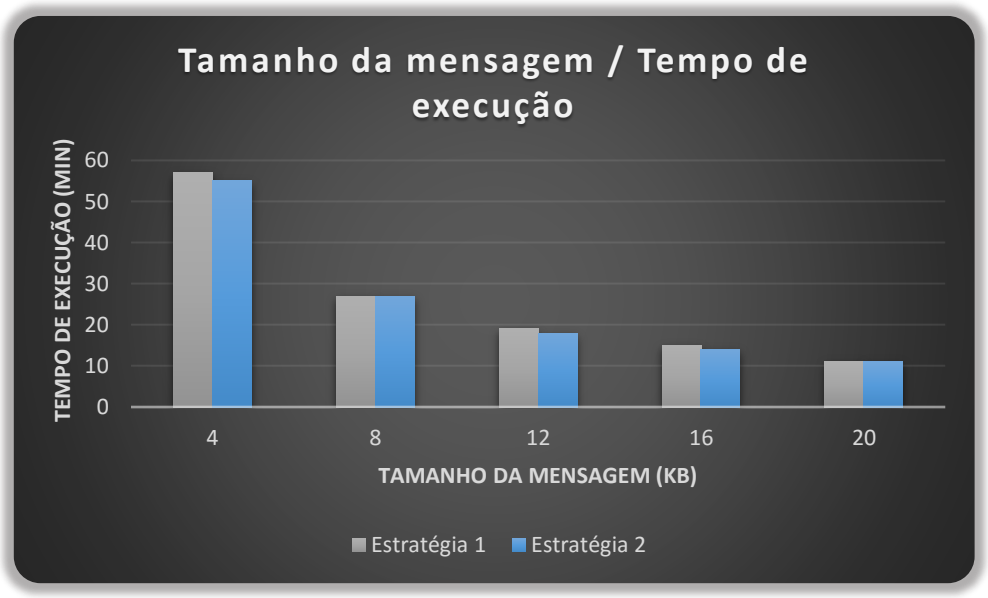


Figura 6.6 – Análise aos resultados alcançados com as duas diferentes estratégias de transmissão (Tamanho da mensagem / Tempo de execução).

Ao contrário do que inicialmente se pudesse pensar, os tempos de execução alcançados na utilização das duas estratégias de transmissão de mensagens são muito próximos. Pelo facto de que na segunda estratégia são efetuadas mais transmissões de mensagens, podia-se pensar que o seu tempo de execução pudesse ser superior em relação à primeira estratégia. Tal não aconteceu e em 3 ocasiões o tempo de execução até foi superior na primeira estratégia, nomeadamente nas experiências com 4, 12 e 16KB. Outra situação que se verifica nos tempos de execução é o facto do mesmo reduzir gradualmente à medida que se vai aumentando o tamanho da mensagem que se utiliza nas experiências. As duas figuras seguintes apresentam essa redução gradual acompanhada paralelamente com a redução do número de mensagens que são transmitidas, isto à medida que se aumenta o tamanho da mensagem nas experiências. Cada uma das figuras seguintes corresponde a uma estratégia de transmissão de mensagens diferente.

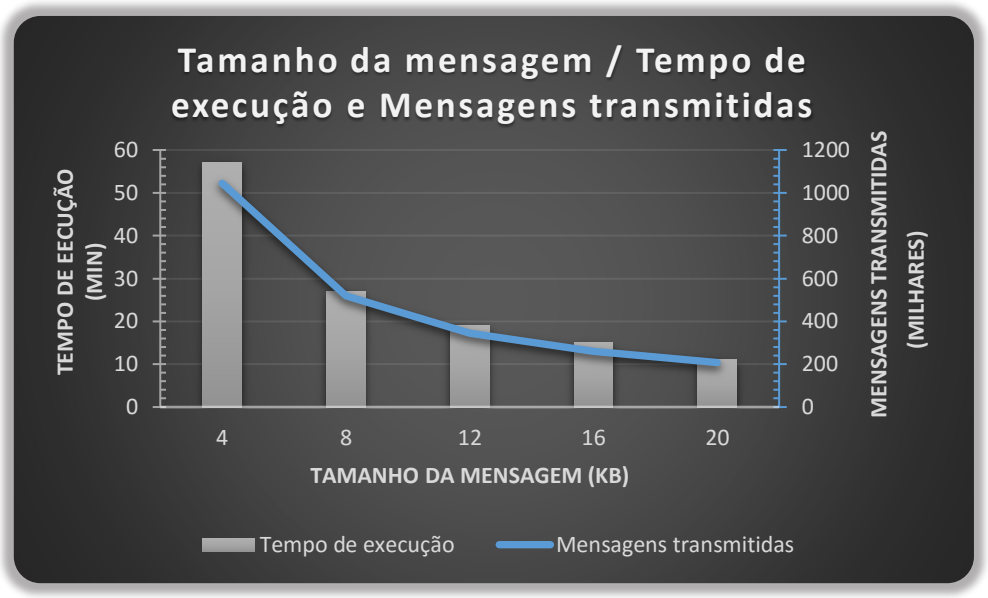


Figura 6.7 – Análise aos resultados alcançados com as duas diferentes estratégias de transmissão (Tamanho da mensagem / Tempo de execução e Mensagens transmitidas – Primeira estratégia de transmissão de mensagens).

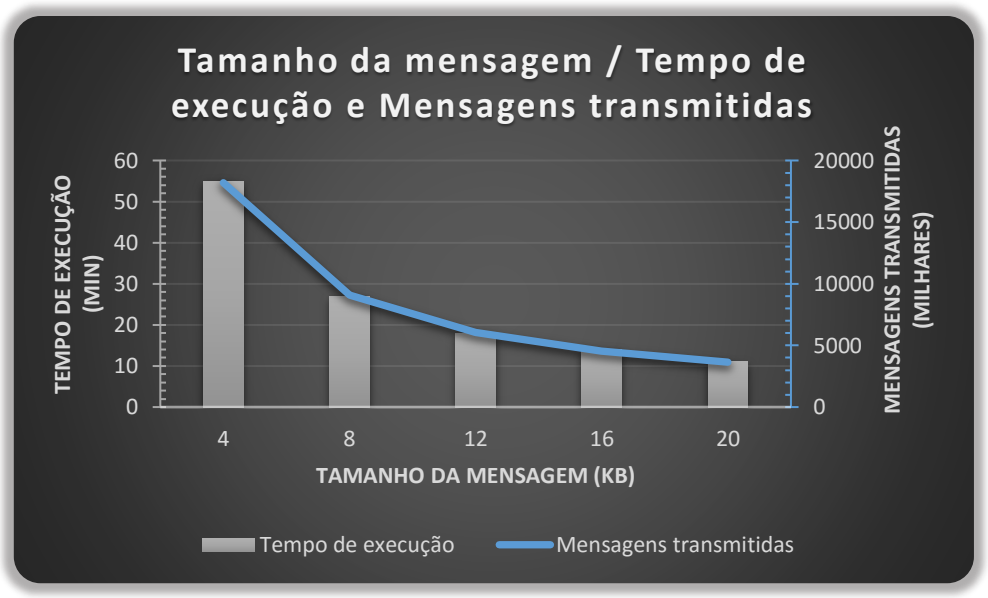


Figura 6.8 – Análise aos resultados alcançados com as duas diferentes estratégias de transmissão (Tamanho da mensagem / Tempo de execução – Segunda estratégia de transmissão de mensagens).

Como é possível visualizar nas figuras anteriores, à medida que se aumenta o tamanho das mensagens nas experiências o número de mensagens transmitidas e o tempo de execução do algoritmo reduz gradualmente. Esta situação é perfeitamente normal, pois com mensagens maiores, maiores serão os slots para as transmissões de mensagens e existirá uma menor quantidade destes slots na respetiva experiência. Apesar do tempo de execução reduzir gradualmente à medida que também são feitas menos transmissões, como se viu anteriormente, estas duas variáveis não aparentam ter dependência, uma vez que com uma menor quantidade de mensagens transmitidas na primeira estratégia, os tempos verificados são muito semelhantes e em alguns casos menores relativamente aos da segunda estratégia. Assim, constata-se que ambas as variáveis referidas dependem do tamanho do slot de transmissão de mensagens, que nestas experiências varia com a variação do tamanho da mensagem.

CAPÍTULO 7
CONSIDERAÇÕES FINAIS E TRABALHO FUTURO

Com este trabalho pretendeu-se dar um contributo para o estudo do desempenho das redes oportunistas. Através do mesmo, é possível efetuar diversas simulações controladas através da aplicação dos algoritmos desenvolvidos. Os resultados alcançados na validação dos dois primeiros algoritmos desenvolvidos (de geração de *logs* RADIUS sintéticos e de extração de encontros, respetivamente) indicam o bom desempenho dos mesmos, permitindo gerar grandes quantidades de registos (primeiro algoritmo) e extrair grandes quantidades de encontros (segundo algoritmo), ambos num curto espaço de tempo.

O terceiro algoritmo que visa o encaminhamento epidémico de mensagens permitiu a extração dos principais resultados inerentes à avaliação de desempenho de uma rede oportunista. A partir dos resultados alcançados neste algoritmo foi possível constatar-se que de facto as DTNs caracterizam-se pelas suas baixas taxas de entrega de mensagens únicas. Apesar disso, face ao protocolo aplicado nas experiências realizadas, esperavam-se taxas mais elevadas. Por outro lado, há que ter em consideração o baixo espaço temporal utilizado (cerca de 8 minutos) que pode levar à redução das referidas taxas, uma vez que pode não existir tempo suficiente para entregar a maioria das mensagens geradas.

Como era expectável, as taxas de entrega foram sempre mais altas quando foi utilizada a segunda estratégia de transmissão de mensagens. Desta forma, foi possível concluir que a taxa de entrega a alcançar é influenciada pela estratégia de transmissão de mensagens que se adote. Por outro lado, através dos resultados obtidos foi também possível concluir que a variação do tamanho da mensagem não se reflete na taxa de entrega alcançada.

Relativamente ao protocolo de encaminhamento aplicado, pôde confirmar-se que de facto o protocolo epidémico requer uma enorme quantidade de recursos, dada à excessiva quantidade de transmissões que efetua e ao considerável espaço de armazenamento que as stations necessitam ter. Considerando que as stations são nós móveis e que possuem baterias relativamente limitadas é de esperar que muitas delas fiquem sem energia e que a qualquer momento deixem de estar ativas na rede. Esta situação acontecerá com maior rapidez quando se utilizar a segunda estratégia de transmissão de mensagens, uma vez que é com esta estratégia que se efetuam mais transmissões. Outra conclusão que se retirou e que vai de encontro ao

esperado, foi o facto de quanto maior for o slot para a transmissão de mensagens (refletido nas experiências através do aumento do tamanho da mensagem) menos transmissões são efetuadas.

No que diz respeito ao atraso, apesar das DTNs serem caracterizadas pelos seus atrasos longos e variáveis e de serem tolerantes aos mesmos, não é de menosprezar a tentativa de se alcançar valores baixos. Os valores médios de atraso registados nas experiências realizadas vão ao encontro desse facto e podem ser considerados aceitáveis face ao espaço temporal da lista de encontros que foi considerado. Esses valores foram de forma clara sempre superiores quando foi utilizada a segunda estratégia de transmissão de mensagens. Desta forma, é possível constatar que o atraso médio depende da estratégia de transmissão de mensagens que se utiliza. Por outro lado, é igualmente possível constatar que não existe uma dependência em relação ao tamanho da mensagem, uma vez que não existe uma tendência visível nos resultados com a variação do tamanho da mensagem.

Relativamente a trabalho futuro, o primeiro passo que poderá ser realizado nesse sentido é a aplicação de dados reais no algoritmo de encaminhamento epidémico desenvolvido neste trabalho. Seguidamente à utilização de dados reais poderá ser feita uma afinação no referido algoritmo, sobretudo no processo de transmissão de mensagens entre as stations. Esta afinação visa tentar melhorar os resultados alcançados, nomeadamente aumentar a taxa de entrega única e reduzir ainda mais o número de transmissões necessárias para tal, de forma a minimizar a energia gasta pelas stations nas suas transmissões. De igual forma, poderá trabalhar-se no sentido de melhorar a gestão das filas de mensagens das stations, através da definição e aplicação de outras formas de gerir este tipo de filas. Algo que no futuro também poderá ser aplicado no algoritmo de encaminhamento epidémico desenvolvido neste trabalho são as sessões de anti-entropia abordadas na literatura (Vahdat e Becker, 2000).

Outra questão que poderá ser realizada no futuro é o desenvolvimento de algoritmos de encaminhamento que apliquem o processo de outros protocolos de encaminhamento, para além do epidémico aplicado neste trabalho. Através disto, é possível efetuar-se diversas comparações entre os resultados alcançados com os diversos protocolos e verificar, por exemplo, qual o que apresenta melhor desempenho.

BIBLIOGRAFIA

Andel, T. R. and Yasinsac, A. (2006). On the credibility of MANET simulations. *Computer*, Vol. 39(7), 48–54, IEEE Computer Society Press Los Alamitos, CA, USA.

Aschenbruck, N.; Munjal, A.; Camp, T. (2011). Trace-based mobility modeling for multi-hop wireless networks. *Computer Communications*, Vol. 34(6), 704–714, Elsevier Science Publishers B. V., Amsterdam, Netherlands.

Barr, R.; Haas, Z. J.; van Renesse, R. (2005). JIST: an efficient approach to simulation using virtual machines: Research Articles. *Software Practice & Experience,* Vol. 35(6), 539–576, John Wiley & Sons, Inc., New York, NY, USA.

Boldrini, C. and Passarella, A. (2010). HCMM: Modeling Spatial and Temporal Properties of Human Mobility Driven by Users' Social Relationships. In *Computer Communications,* Vol. 33(9), 1056–1074, Elsevier Science Publishers B.V., Amsterdam, Netherlands.

Boldrini, C.; Conti, M.; Jacopini, J.; Passarella, A. (2007). HiBOp: a history based routing protocol for opportunistic networks. In *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2007)*, 1–12.

Boldrini, C.; Conti, M.; Passarella, A. (2008). Exploiting users' social relations to forward data in opportunistic networks: The HiBOp solution. *Pervasive and Mobile Computing*, Vol. 4 (5), 633–657, Elsevier Publishing Solutions B.V., Amsterdam, Netherlands.

Boldrini, C.; Conti, M.; Passarella A.; Karamshuk, D. (2011). Human Mobility Models in Opportunistic Networks. In *IEEE Communications Magazine*, Vol. 49(12), 157-165.

Borrel, V.; Legendre, F.; de Amorim, M. D.; Fdida, S. (2009). SIMPS: Using Sociology for Personal Mobility. In *IEEE/ACM Transactions on Networking (TON)*, Vol. 17(3), 831–842, IEEE Press Piscataway, NJ, USA.

Burgess, John; Gallagher, Brian; Jensen, David; Levine, Brian Neil. (2006). MaxProp: routing for vehicle-based disruption-tolerant networks. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, 1-11.

Campbell, Andrew T.; Eisenman, Shane B.; Lane, Nicholas D.; Miluzzo, Emiliano; Peterson, Ronald A. (2006). People-centric urban sensing. In *Proceedings of the 2nd annual international workshop on Wireless internet (WICON '06),* Article 18, ACM, New York, NY, USA.

Camp, T.; Boleng, J.; Davies, V. (2002). A survey of mobility models for ad hoc network research. In *Wireless Communication and Mobile Computing Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, Vol. 2(5), 483–502.

Calegari, R.; Musolesi, M.; Raimondi, F.; Mascolo, C. (2007). CTG: a connectivity trace generator for testing the performance of opportunistic mobile systems. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering (ESEC-FSE '07)*, 415-424, ACM, New York, NY, USA.

Chaintreau, A.; Hui, P.; Crowcroft, J.; Diot, C.; Gass, R.; Scott, J. (2006). Impact of human mobility on the design of opportunistic forwarding algorithms. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM 2006),* 1-13, Barcelona, Spain.

Chaintreau, A.; Hui, P.; Crowcroft, J.; Diot, C.; Gass, R.; Scott, J. (2005). Pocket switched networks: Real-world mobility and its consequences for opportunistic forwarding. Technical Report, UCAM-CL-TR-617, University of Cambridge, Computer Laboratory.

Chaintreau, A.; Mtibaa, A.; Massoulie, L.; Diot, C. (2007). The diameter of opportunistic mobile networks. In *Proceedings of the 3rd International Conference on Emerging Networking Experiments and Technologies (CONEXT '07)*, Article 12, 1-12, ACM, New York, NY, USA.

Coombs, Richard (2014). *Models for information propagation in opportunistic networks.* PhD Thesis, Cardiff University, United Kingdom.

Daly, E. and Haahr, M. (2007). Social network analysis for routing in disconnected delay-tolerant MANETs. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc '07)*, 32-40, ACM, New York, NY, USA.

Fischer, D.; Herrmann, K.; Rothermel, K. (2010). GeSoMo — A general social mobility model for delay Tolerant networks. In *Mobile Adhoc and Sensor Systems (MASS), 2010 IEEE 7th International Conference on*, 99–108, IEEE, San Francisco, CA, USA.

Grasic, S. and Lindgren, A. (2012). An analysis of evaluation practices for DTN routing protocols. In *Proceedings of the 7th ACM international workshop on Challenged networks (CHANTS '12)*, 57-64, ACM, New York, NY, USA.

Hsu, W. J.; Merchant, K.; Shu, H.; Hsu, C.; Helmy, A. (2005). Weighted waypoint mobility model and its impact on ad hoc networks. In *Mobile Computing and Communication Review*, Vol. 9(1), 59-63, ACM, New York, NY, USA.

Heinemann, A.; Kangasharju, J.; Muhlhauser, J. (2008). Opportunistic data dissemination using real-world user mobility traces. In *Proceedings of 22nd International Conference on Advanced Information Networking and Applications—Workshops (AINAW 2008)*, 1715-1720, IEEE Computer Society, Washington, DC, USA.

Helgason, O. R. and Jonsson, K. V. (2008). Opportunistic networking in OMNeT++. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops (Simutools '08*), Article 82, 1-8, ICST, Brussels, Belgium.

Henderson, T.; Kotz, D.; Abyzov, I. (2004). The changing usage of a mature campus-wide wireless network. In *Proceedings of the 10th annual international conference on Mobile computing and networking (MobiCom '04)*, 187–201, ACM, New York, NY, USA.

Henderson, T. R.; Roy, S.; Floyd, S.; Riley, G. F. (2010). *ns-3* project goals. In *Proceeding from the 2006 workshop on ns-2: the IP network simulator (WNS2 '06)*, Article 13, ACM, New York, NY, USA.

Hui, P.; Crowcroft, J.; Yoneki, E. (2011). BUBBLE Rap: Social-Based Forwarding in Delay-Tolerant Networks. *IEEE Transactions on Mobile Computing*, Vol. 10(11), 1576–1589, IEEE Computer Society, Washington, DC, USA.

Hsu, Wei-jen and Helmy, Ahmed. (2010). On Nodal Encounter Patterns in Wireless LAN Traces, In *IEEE Transactions on Mobile Computing*, Vol. 9(11), 1563-1577.

Jahromi, K.; Meneses, F.; Moreira, A. (2014). Analyzing temporal scale behaviour of connectivity properties of node encounters. In *Proceedings of the 4th International Conference on Selected Topics in Mobile & Wireless Networking (MoWNet'2014)*, *Procedia Computer Science*, Vol. 40, 57-65, Elsevier Science Publishers B.V., Amsterdam, Netherlands.

Jahromi, K.; Meneses, F.; Moreira, A. (2014). Impact of Ping-Pong Events on Connectivity Properties of Node Encounters. In *Proceedings of the 7th IFIP Wireless and Mobile Networking Conference*, Vilamoura, Portugal.

Jardosh, A.; Belding-Royer, E. M.; Almeroth, K. C.; Suri, S. (2003). Towards realistic mobility models for mobile ad hoc networks. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom'03)*, 217-229, ACM, New York, USA.

Juang, Philo; Oki, Hidekazu; Wang, Yong; Martonosi, Margaret; Peh, Li Shiuan; Rubenstein, Daniel. (2002). Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems (ASPLOS X)*, 96-107, ACM, New York, NY, USA.

Kathiravelu, T. and Pears, A. (2011). State of the Art in Modeling Opportunistic Networks. In M. K. Denko (Ed), *Mobile Opportunistic Networks: Architectures, Protocols and Applications*, (Chapter 2, pp. 26-46). Boca Raton: Auerbach Publications.

Kathiravelu, T. and Pears A. (2006). What & when: Distributing content in opportunistic networks. In *Proceedings of the International Conference on Wireless and Mobile Computing (ICWMC 2006*), 64-64, Bucharest, Romania.

Keränen, A.; Ott, J.; Kärkkäinen, T. (2009). The ONE simulator for DTN protocol evaluation. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques (Simutools '09)*, Article 55, 1-10, ICST, Brussels, Belgium.

Keshav, S.; Chawathe, Y.; Chen, M.; Zhang, Y.; Wolman, A. (2007). Cell phones as a research platform. In *Proceedings of the 5th international conference on Mobile systems, applications and services (MobiSys '07)*, 2-2, ACM, New York, NY, USA.

Khelil, A.; Marrón, P.J.; Rothermel, K. (2005). Contact-based mobility metrics for delay-tolerant ad hoc networking. In *13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 435-444.

Kim, M.; Kotz, D.; Kim, S. (2006). Extracting a mobility model from real user traces. In *INFOCOM 2006*, *25th IEEE Conference on Computer Communications*, 1-13, Barcelona, Spain.

Kim, Sungwon and Eun, Do Young. Impact of Super-Diffusive Behavior on Routing Performance in Delay Tolerant Networks. In *IEEE International Conference on Communications (ICC '08),* 2941-2945.

Konishi, K.; Maeda, K.; Sato, K.; Yamasaki, A.; Yamaguchi, H.; Yasumoto, K.; Hi-gashino. T. (2005). Mobireal simulator - evaluating MANET applications in real environments. *In Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS2005)*, 492-502, IEEE Computer Society, Washington, DC, USA.

Kurkowski, S.; Camp, T.; Colagrosso, M. (2004). MANET Simulation Studies: The Current State and New Simulation Tools.

Kurkowski, S.; Camp, T.; Colagrosso, M. (2008). MANET Simulation Studies: The Incredibles. *SIGMOBILE Mobile Computing and Communications Review*, Vol. 9(4), 50–61, ACM, New York, NY, USA.

LeBrun, J. and Chuah, C. N. (2006). Bluetooth-based content distribution stations on public transit systems. In *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking (MobiShare '06)*, 63-65, ACM, New York, NY, USA.

Leguay, J.; Lindgren, A.; Scott, J.; Friedman, T.; Crowcroft, J. (2006). Opportunistic content distribution in an urban setting. In *Proceedings of the 2006 SIGCOMM workshop on Challenged networks (CHANTS '06*), 205-212, ACM, New York, NY, USA.

Lindgren, Anders; Doria, Avri; Schelén, Olov. (2003). Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol. 7(3), 19-20, ACM, New York, NY, USA.

Li, Q.; Zhu, S.; Cao, G. (2010). Routing in socially selfish delay tolerant networks. In *Proceedings of the 29th conference on Information communications (INFOCOM'10)*, 857-865, IEEE Press, Piscataway, NJ, USA.

Martins, F. Mário. (2009). O Paradigma da Programação Orientada Pelos Objetos. *JAVA6 e Programação Orientada Pelos Objetos*, 4ª Edição, (Capítulo 1, pp. 1-26). Lisboa: FCA.

Mascolo, C. and Musolesi, M. (2006). SCAR: Context-aware Adaptive Routing in Delay Tolerant Mobile Sensor Networks. In *Proceedings of the 2006 international conference on Wireless communications and mobile computing (IWCMC '06)*, 533-538, ACM, New York, NY, USA.

McNett, M. and Voelker, G. M. (2005). Access and mobility of wireless PDA users. *SIGMOBILE Mobile Computing and Communications Review*, Vol. 9(2), 40-55, ACM, New York, NY, USA.

Mota, Vinícius F.S.; Cunha, Felipe D.; Macedo, Daniel F.; Nogueira, José M.S.; Loureiro, António A.F. (2014). Protocols, Mobility models and Tools in Opportunistic Networks: A Survey. *Computer Communications*, Vol. 48, 5-19, Elsevier Publishing Solutions B.V., Amsterdam, Netherlands.

Mtibaa, A.; May, M.; Diot, C.; Ammar, M. (2010). Peoplerank: social opportunistic forwarding. In *Proceedings of the 29th conference on Information communications (INFOCOM'10)*, 111-115, IEEE Press, Piscataway, NJ, USA.

Musolesi, M. and Mascolo, C. (2006). A community-based mobility model for ad hoc network research. In *Proceedings of the 2nd ACM/SIGMOBILE International Workshop on Multi-hop Ad Hoc Networks: From Theory to Reality (REALMAN'06)*, 31-38, ACM, New York, NY, USA.

Musolesi, M. and Mascolo, C. (2009). Car: Context-aware adaptive routing for delay tolerant mobile networks. *IEEE Transactions on Mobile Computing*, Vol. 8(2), 246-260, IEEE Educational Activities Piscataway, NJ, USA.

Musolesi, M. and Mascolo, M. (2007). Designing mobility models based on social network theory. *SIGMOBILE Mobile Computing and Communications Review*, Vol. 11(3), 59–70, ACM, New York, NY, USA.

Musolesi, M.; Hailes, S.; Mascolo, C. (2004). An ad hoc mobility model founded on social network theory. In *Proceedings of the 7th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems* (MSWiM '04), 20-24, ACM, New York, NY, USA.

Perkins, C. E. and Bhagwat, P. (1994). Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. In *Proceedings of the conference on Communications architectures, protocols and applications (SIGCOMM '94)*, 234-244, ACM, New York, NY, USA.

Perkins, C. E. and Royer, E. M. (1999). Ad-hoc on-demand distance vector routing. In *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications (WMCSA '99)*, 90–90, IEEE Computer Society, Washington, DC, USA.

Radenkovic, M. and Grundy, A. (2011). Framework for utility driven congestion control in delay tolerant opportunistic networks. In *Proceedings of the 7th International IEEE Wireless Communications and Mobile Computing Conference (IWCMC '11)*, 448 –454, Istanbul, Turkey.

Ristanovic, N.; Theodorakopoulos, G.; Le Boudec, J-Y. (2012). Traps and pitfalls of using contact traces in performance studies of opportunistic networks. In *Proceedings of the IEEE INFOCOM 2012*, 1377–1385.

Song, Libo and Kotz, David F. Evaluating opportunistic routing protocols with large realistic contact traces. In *Proceedings of the 2nd ACM workshop on Challenged networks (CHANTS '07)*, 35–42, ACM, New York, NY, USA.

Spyropoulos, T.; Psounis, K.; Raghavendra, C. (2004). Single-copy routing in intermittently connected mobile networks. In *Proceedings of the 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (IEEE SECON 2004)*, 235-244.

Spyropoulos, T.; Psounis, K.; Raghavendra, C. (2005). Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking (WDTN '05)*, 252-259, ACM, New York, NY, USA.

Su, J.; Chin, A.; Popivanova, A.; Goel, A.; de Lara, E. (2004). User mobility for opportunistic ad-hoc networking. In *Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '04)*, 41-50, IEEE Computer Society, Washington, DC, USA.

Su, Jing; Goel, Ashvin; de Lara, Eyal. (2006). An empirical evaluation of the student-net delay tolerant network. In *Proceedings of the 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services (MOBIQUITOUS 2006)*, 1-10.

Vahdat, Amin and Becker, David. (2000). Epidemic routing for partially-connected ad hoc networks. *Technical Report CS-2000-06*, Duke University.

Varga, A. and Hornig, R. (2008). An overview of the OMNeT++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops (Simutools '08*), Article 60, 1-10, ICST, Brussels, Belgium.

Varga, A. (2001). The OMNeT++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM'2001),* Prague.

Vu, L.; Do, Q.; Nahrstedt, K. (2011). 3R: fine-grained encounter-based routing in delay tolerant networks. In *Proceedings of the 2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WOWMOM '11)*, 1-6, IEEE Computer Society, Washington, DC, USA.

Whitbeck, J. and Conan, V. HYMAD: Hybrid DTN-MANET routing for dense and highly dynamic wireless networks. In *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks & Workshops (WoWMoM 2009)*, 1-7.

Xie, X.; Zhang, Y.; Dai, C.; Song, M. (2011). Social relationship enhanced predicable routing in opportunistic network. In *Proceedings of the 2011 Seventh International Conference on Mobile Ad-hoc and Sensor Networks (MSN '11)*, 268-275, IEEE Computer Society, Washington, DC, USA.

Yang, S.; Yang, X.; Zhang, C.; Spyrou, E. (2010). Using social network theory for modeling human mobility. In *IEEE Network: The Magazine of Global Internetworking*, Vol. 24(5), 6 –13, IEEE Press Piscataway, NJ, USA.

Yeo, J.; Kotz, D.; Henderson, T. (2006). CRAWDAD: a community resource for archiving wireless data at Dartmouth. *SIGCOMM Computer Communication Review*, Vol. 36(2), 21–22, ACM, New York, NY, USA.

Zheng, Y.; Zhang, L.; Xie, X.; Ma, W. (2009). Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the 18th international conference on World wide web (WWW '09)*, 791-800, ACM, New York, NY, USA.

Zyba, G.; Voelker, G.; Ioannidis, S.; Diot, C. (2011). Dissemination in opportunistic mobile ad-hoc networks: the power of the crowd. In *Proceedings of the IEEE INFOCOM 2011*, 1179–1187.

ANEXOS

ANEXO 1 - Implementação algoritmo para geração de *logs* RADIUS sintéticos

PACKAGE DATABASE

Como pode ser visto anteriormente na figura 4.2, o package database inclui duas classes, uma relativa à ligação à base de dados, *ConnectionDatabase*, e outra relativa aos processos de manipulação dos dados da mesma, *DataAccess*.

→ Classe *ConnectionDatabase*

Basicamente, o conteúdo da classe que faz a ligação à base de dados pode ser dividida em 4 partes principais:

- Identificação do nome do controlador JDBC e do endereço da base de dados;
- Identificação das credenciais da base de dados;
- Registo do controlador JDBC;
- Abertura de uma conexão.

O controlador JDBC é um conjunto de classes e interfaces escritas em JAVA que fazem o envio de instruções SQL para qualquer base de dados relacional. É definido pela Sun Microsystems e é implementado através de interfaces padrão java.sql.

→ Classe *DataAccess*

A classe *DataAccess* corresponde aos processos que necessitem de aceder à base de dados, para qualquer manipulação de dados. Neste algoritmo, o único processo inerente a esta classe é o da escrita dos registos gerados na base de dados.

Este processo de escrita inclui o seguinte método:

- ***sendRecordsToDatabase()*:** este método inclui uma instrução SQL que permite inserir na base de dados os registos que são gerados após uma simulação, mais especificamente na tabela dos registos.

PACKAGE GENERATOR

Relativamente ao package generator, este é composto por uma classe principal denominada *RadiuslogEventGenerator* e outras três representando os objetos (instâncias de classe) *AP*, *Station* e *Event*.

→ Classe ***RadiuslogEventsGenerator***

Como foi dito anteriormente, a classe *RadiuslogEventGenerator* é a classe principal, onde está incluído o método *main()*. É a partir desta classe que os três objetos são instanciados, garantindo que sempre que sejam criados tenham igual estrutura e comportamento.

Esta classe contém outros quatro métodos que são declarados fora do método main, sendo invocados dentro deste para serem executados.

- ***getUserInput()*:** permite ao utilizador inserir valores para os parâmetros da simulação/entrada, ou seja, o número de APs, o raio de cobertura dos mesmos, o número de Stations e por fim o tempo de simulação (tf);
- ***initialization()*:** permite inicializar os APs e as stations;
- ***generateRecords()*:** permite gerar os registos, principal objetivo do algoritmo;
- ***writeToDatabase()*:** permite escrever os registos gerados pelo método anterior na base de dados.

→ Classe ***AP***

A classe *AP* refere-se ao objeto AP.

O objeto AP possui as seguintes variáveis de instância:

- **apID:** id do AP;
- **x:** posição do AP na área de simulação, eixo do x;
- **y:** posição do AP na área de simulação, eixo do y.

E os seguintes métodos de instância:

- ***setPosition():*** permite distribuir aleatoriamente cada AP pela área de simulação;
- ***distanceToAP():*** permite calcular a distância da station a cada AP.

Cada uma das variáveis de instância possuem os respetivos métodos get() e set().

→ **Classe *Station***

À semelhança da classe anterior, a classe *Station* refere-se a um objeto, neste caso ao objeto Station.

O objeto Station possui as seguintes variáveis de instância:

- **stationID:** id da station;
- **x:** posição da station na área de simulação, eixo do x;
- **y:** posição da station na área de simulação, eixo do y;
- **t:** instante de tempo;
- **ap:** o id do AP que está mais próximo da station.

E os seguintes métodos de instância:

- ***initStation()*:** permite distribuir aleatoriamente cada station pela área de simulação;
- ***distanceToAP()*:** permite calcular a distância da station a cada AP;
- ***getAssociatedAP()*:** permite obter o AP que está mais próximo à station, verificando se a mesma está ou não dentro da área de cobertura do respetivo AP. Este método retorna um AP, o AP mais próximo, apenas quanto a station está dentro da área de cobertura do mesmo, em caso contrário não retorna qualquer valor;
- ***stationMov()*:** permite mover aleatoriamente a station pela área de simulação após ser inicializada.
- ***nearestAP()*:** permite verificar qual a posição do AP mais próximo à station após a mesma ter-se movimentado aleatoriamente na área de simulação.

Cada uma das variáveis de instância possui os respetivos métodos get() e set().

→ **Classe *Event***

A classe *Event* representa o objeto Event. Este objeto corresponde estruturalmente a um registo.

O objeto Event possui as seguintes variáveis de instância:

- **t:** instante de tempo;
- **s:** id da station;
- **ap:** id do AP;
- **e:** tipo de evento, ou seja, *start* ou *stop*.

Cada uma destas variáveis de instância possui os respetivos métodos get() e set().

ANEXO 2 - Implementação do algoritmo para extração de encontros

PACKAGE DATABASE

O *package database* engloba três classes: uma que permite efetuar a ligação à base de dados, denominada *ConnectionDatabase*, outra que permite aceder e manipular os dados da base de dados, denominada DataAccess e uma terceira denominada *Record*, que corresponde ao objeto Record (registo) que será utilizado como parâmetro de entrada neste algoritmo e acedido através da base de dados.

→ Classe *ConnectionDatabase*

Esta classe, responsável pela ligação à base de dados, está descrita na implementação do algoritmo anterior, tendo exatamente a mesma finalidade e denominação da classe desse algoritmo.

→ Classe *DataAccess*

Esta classe também está presente no algoritmo anterior e como foi dito corresponde a todos os processos que necessitem de aceder à base de dados para qualquer manipulação de dados. Especificamente neste algoritmo, existem inúmeros processos que necessitam de tal procedimento.

Esta classe possui os seguintes métodos:

- ***getUniqueAPList()*:** permite obter a lista de APs únicos da base de dados, mais especificamente da tabela dos registos;
- ***getNumberAPs()*:** permite obter o número de APs únicos existentes nessa mesma tabela de registos;
- ***getNumberRecordsAP()*:** permite obter da tabela de registos da base de dados, o número de registos de cada AP único;
- ***getRecords()*:** permite obter da mesma tabela anterior, os registos associados a cada AP único;
- ***sendRecordsToDatabase()*:** permite enviar para a base de dados, mais especificamente para a tabela dos encontros, os encontros gerados pela execução do algoritmo.

→ **Classe *Record***

Como foi dito anteriormente, esta classe representa o objeto *Record*.

Os registos presentes na base de dados e mais especificamente na tabela dos registos são utilizados neste algoritmo como parâmetro de entrada, sendo como é logico indispensáveis para o processo de extração de encontros.

O objeto *Record* possui as seguintes variáveis de instância:

- **t:** instante de tempo associado ao registo;
- **stationID:** id da station associada ao registo;
- **eventType:** tipo do evento/registo, ou seja, START ou STOP.

PACKAGE ENCOUNTERS

Relativamente ao package *encounters*, este é composto por uma classe principal denominada *EncountersDetection* e uma outra que representa um objeto (instância de classe) denominada *Encounter*.

→ **Classe *EncountersDetection***

A classe *EncountersDetection* é a classe principal, onde está incluído o método main().

Esta classe contém outros dois métodos que são declarados fora do método main(), sendo invocados dentro deste para serem executados.

- ***generateEncounters()*:** método onde está presente o processo lógico da geração dos encontros e que permite gerar os encontros;
- ***writeToDatabase()*:** método que invoca o método *sendRecordsToDatabase()* da classe *DataAccess*, permitindo escrever os encontros gerados pelo método anterior na base de dados, mais especificamente na tabela dos encontros.

→ **Classe *Encounter***

A classe *Encounter* representa o objeto *Encounter* (encontro).

Cada objeto deste tipo que for criado terá sempre o mesmo formato apresentado nesta classe.

O objeto *Encounter* possui as seguintes variáveis de instância:

- **id:** id do encontro, que será automaticamente incrementado na base de dados;
- **apID:** id do AP associado ao encontro;
- **stationA_ID:** id de uma das stations associadas ao encontro;
- **stationB_ID:** id da outra station associada ao encontro;
- **t1:** instante de tempo relativo ao início do encontro (*start_time*);
- **t2:** instante de tempo relativo ao fim do encontro (*end_time*).

Cada uma destas variáveis possuem os respetivos métodos get() e set

ANEXO 3 - Implementação do algoritmo para o encaminhamento epidémico de mensagens

PACKAGE DATABASE

O package database é composto por duas classes: uma semelhante à dos anteriores algoritmos, denominada *ConnectionDatabase*, que permite efetuar a ligação à base de dados e uma outra que permite aceder e manipular os dados da base de dados, denominada *DataAccess*.

→ **Classe *ConnectionDatabase***

Esta classe, responsável pela ligação à base de dados, está descrita no anexo 1 referente à implementação do primeiro algoritmo, tendo exatamente a mesma finalidade e denominação da classe desse algoritmo.

→ **Classe *DataAccess***

Esta classe está presente nos algoritmos anteriores e corresponde a todos os processos que necessitem de aceder à base de dados para qualquer manipulação de dados. Especificamente para este algoritmo foram implementados métodos específicos, de forma atender as necessidades específicas do seu processamento.

Sendo assim, esta classe possui os seguintes métodos:

- ***getUniqueStations():*** permite obter os ids das stations únicas que existem na lista de encontros;
- ***getNumberOfStations():*** permite obter o número de stations únicas que existem no *trace* de registos RADIUS;
- ***getNeighbors():*** permite obter os vizinhos de uma dada station num determinado instante de tempo;
- ***getUniqueEventsList():*** permite obter uma lista de eventos com tempos únicos no formato "tempo, tipo de evento". Esta lista está ordenada pelo tempo;
- ***getEncounters():*** permite obter todos os encontros;
- ***getEncountersStartTime(int eventTime):*** permite obter os encontros que se iniciem em eventTime;

- ***getEncountersEndTime(int eventTime):*** permite obter os encontros que terminem em eventTime;
- ***getUniqueAps(int time):*** permite obter uma lista com os aps únicos;
- ***getStationsEachAp(int time, int ap):*** permite obter a lista de stations que estão associadas a um ap num determinado instante de tempo;
- ***getTime():*** permite obter os instantes de tempo no formato inteiro;
- ***getDate():*** permite obter os instantes de tempo no formato data;
- ***saveResults():*** guarda os resultados da execução do algoritmo na base de dados.

PACKAGE ENTITIES

O package entities é composto por cinco classes que representam os objetos (instâncias de classe): Encounter (representa um encontro), Message (representa uma mensagem), *Station* (representa uma station), Event (representa um evento) e por fim a classe Neighbor (representa um vizinho).

→ **Classe *Encounter***

À semelhança do que foi explicado no anexo 2 referente ao segundo algoritmo, a classe *Encounter* representa o objeto Encounter (encontro).

Cada objeto deste tipo que for criado terá sempre o mesmo formato apresentado nesta classe.

O objeto *Encounter* possui as seguintes variáveis de instância:

- **id:** id do encontro, que será automaticamente incrementado na base de dados;
- **apID:** id do AP associado ao encontro;
- **stationA_ID:** id de uma das stations associadas ao encontro;
- **stationB_ID:** id da outra station associada ao encontro;
- **t1:** instante de tempo relativo ao início do encontro (*start_time*);
- **t2:** instante de tempo relativo ao fim do encontro (*end_time*).

Cada uma destas variáveis possuem os respetivos métodos get() e set().

→ **Classe *Message***

A classe *Message* representa o objeto Message (mensagem).

Como foi explicado anteriormente para a classe anterior, cada objeto deste tipo que for criado terá sempre o mesmo formato apresentado nesta classe.

O objeto Message possui as seguintes variáveis de instância:

- **id:** id da mensagem;
- **message:** conteúdo da mensagem
- **sourceStation:** id da station que gerou a mensagem;
- **destinationStation:** id da station de destino da mensagem;
- **ttl:** tempo de vida da mensagem.

Para além dos respetivos métodos get() e set(), esta classe tem o seguinte método de instância adicional:

- ***messageIsValid()*:** permite verificar a validade da mensagem.

→ **Classe *Station***

À semelhança da classe anterior, a classe *Station* refere-se a um objeto, neste caso ao objeto Station. Para além disso, esta classe tem uma característica especial, pois herda todas as variáveis e métodos da classe *Neighbor*, referente ao objeto *Neighbor*.

O objeto Station possui as seguintes variáveis de instância:

- **id:** id da station;
- **outputBuffer:** fila de saída de mensagens da station;
- **outputBufferCapacity:** capacidade da fila de saída de mensagens da station.

E os seguintes métodos de instância:

- ***generateMessage()*:** permite gerar uma mensagem a partir de uma determinada station ativa que está a ser processada;
- ***transmitMessage()*:** permite transmitir uma mensagem de uma dada station (mensagem guardada na fila de saída da respetiva station) para outra station (guardando essa mensagem na fila de saída dessa station).
- ***getNeighbors()*:** permite obter todos os vizinhos de uma determinada station;

- ***getNeighbor()*:** permite obter um vizinho especifico de uma determinada station;
- ***compareStations()*:** permite comparar a station que está a ser processada com as stations ativas (activeStations(, de forma a verificar se já é uma dessas stations.

Cada uma das variáveis de instância possui os respetivos métodos get() e set().

→ Classe *Event*

A classe *Event* representa o objeto Event.

O objeto Event possui as seguintes variáveis de instância:

- **t:** instante de tempo;
- **eventType:** tipo de evento (start ou stop).

Cada uma destas variáveis de instância possui os respetivos métodos get() e set().

→ Classe *Neighbor*

A classe *Neighbor* representa o objeto Neighbor (vizinho).

O objeto Neighbor possui as seguintes variáveis de instância:

- **listOfNeighbors:** lista de vizinhos;
- **currentNeighborPointer:** variável que percorre a lista de vizinhos, permitindo verificar a qualquer momento para que vizinho está "apontar".

E os seguintes métodos de instância:

- ***getNextNeighbor()*:** permite retornar o próximo vizinho para onde será transmitida a próxima mensagem. O próximo vizinho será a station para onde está "apontar" a variável currentNeighborPointer na lista de vizinhos da station que vai transmitir;
- ***addNeighbor():*** permite adicionar um vizinho à lista de vizinhos.
- ***sortNeighbors()*:** permite ordenar a lista de vizinhos de cada station ativa.

PACKAGE EPIDEMIC

Relativamente ao package epidemic, este é composto por uma classe principal denominada *epidemicRouting* e uma outra denominada de runningEncounters que engloba todos os encontros que estão em curso e respetivos processos associados aos mesmos.

→ **Classe *EpidemicRouting***

A classe *EpidemicRouting* é a classe principal, onde está incluído o método main().

Esta classe contém outros dois métodos que são declarados fora do método main(), sendo invocados dentro deste para serem executados.

- ***getUserChoice()*:** permite obter a escolha do utilizador para os parâmetros de entrada do algoritmo e entre as duas estratégias de transmissão de mensagens disponíveis;
- ***processEncounters()*:** permite processar toda a lista de encontros, onde estão incluídos os processos de atualizar os encontros em curso e as stations ativas nesses encontros e gerar e transmitir mensagens através dessas stations ativas.

→ **Classe *RunningEncounters***

A classe *RunningEncounters* corresponde a todos os processos ligados aos encontros que estão em curso. De forma a uma melhor perceção desta classe, seguidamente são apresentadas as variáveis incluídas na mesma:

- **activeEncounters:** lista dos encontros que estão em curso;
- **activeStations:** lista de stations ativas (presentes em activeEncounters);
- **generaredMessageCounter:** contador de mensagens geradas;
- **deliveredMessageCounter:** contador de mensagens entregue ao seus destino;
- **deliveredUniqueMessageCounter:** contador de mensagens únicas entregues ao seu destino;
- **queueLenghtMap():** cumprimento das filas de mensagens de cada station;
- **delay:** lista onde são guardados os valores do atraso;
- **messages:** lista de mensagens;
- **uniqueAps:** lista de aps únicos;
- **stationsEachAp:** lista com as stations associadas a cada ap.
- **transmissionStrategy:** estratégia de transmissão de mensagens

Esta classe possui os seguintes métodos principais:

- ***updateActiveEncounters*:** permite atualizar a lista de encontros ativos, adicionando (***addEncounters()***) ou removendo (***removeEncounters()***) encontros da mesma;
- ***updateActiveStations()*:** permite atualizar a lista de stations ativas, verificando se as stations que estão em activeEncounters já estão em activeStations;
- ***updateUniqueAps(Event event)*:** atualiza os aps únicos existentes num determinado evento;
- ***updateStationsEachAp(int time, int ap)*:** permite atualizar a lista de stations associadas a um determinado ap num dado instante de tempo;
- ***generateMessages()*:** permite percorrer todas as stations ativas e para cada uma gera ou não uma mensagem. Caso tenha que gerar uma mensagem é invocado o método ***generateMessage()*** da classe station;
- ***transmitFromNextStation()*:** permite que uma station ativa transmita por cada slot de tempo (transmitSlot). Este método corresponde à primeira estratégia de transmissão de mensagens que o utilizador pode escolher no início da simulação;
- ***transmitFromAllStations()*:** permite que cada uma das stations ativas tenha oportunidade de transmitir uma mensagem em cada slot de tempo (transmitSlot). Este método corresponde à segunda estratégia de transmissão de mensagens que o utilizador pode escolher no início da simulação;
- ***getAverageDelay()***, ***getMinDelay()*** e ***getMaxiDelay()*:** permite obter respetivamente o atraso médio, mínimo e máximo;
- ***queueLenghtAtEnd():*** permite verificar o cumprimento das filas de mensagens de cada station no final da simulação;
- ***round(double value, int scale):*** permite arredondar valores (resultados, por exemplo) utilizando uma determinada escala;
- ***showResults():*** permite mostrar resultados da simulação, de acordo com as métricas utilizadas e especificadas na abordagem metodológica (Capítulo 3, secção 3.2).

PACKAGE QUEUE

Relativamente ao package queue, este é composto por uma única classe denominada *ArrayQueue* que engloba toda a gestão de cada fila de mensagens.

→ Classe *ArrayQueue*

A classe *ArrayQueue* representa a fila de mensagens de cada station.

Esta classe possui as seguintes variáveis:

- **queue:** fila de mensagens;
- **size:** variável que percorre a lista de vizinhos, permitindo verificar a qualquer momento para que vizinho está "apontar";
- **first:** primeira posição da fila;
- **last:** última posição da fila;
- **capacity:** capacidade da fila de mensagens;
- **currentMessagePointer:** variável que percorre a lista de mensagens, permitindo verificar a qualquer momento para que mensagem está "apontar".
- **isDelivered:** esta variável é um *boolean* que representa se a mensagem foi ou não entregue.

E os seguintes métodos de instância:

- ***ArrayQueue(int capacity)*:** permite criar uma nova fila de mensagens com uma determinada capacidade;
- ***size()*:** permite obter o tamanho da fila de mensagens;
- ***isEmpty()*:** permite verificar se a fila está vazia (size = 0);
- ***peekLast()*:** permite obter a última mensagem da fila;
- ***addMessage(Message message, int time)*:** permite adicionar uma dada mensagem a uma fila num determinado instante de tempo;
- ***notExistsMessage(Message message)*:** permite verificar se uma dada mensagem já existe numa determinada fila;
- ***orderArrayQueue()*:** permite ordenar uma fila de mensagens por ordem crescente do ttl de cada mensagem;
- ***removeMessage()*:** permite remover a mensagem presente na última posição da fila;