**UNIVERSIDADE FEDERAL DE SÃO CARLOS**

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# APRIMORANDO O DESEMPENHO DE ALGORITMOS DE ROTEAMENTO EM VANETS UTILIZANDO CLASSIFICAÇÃO

**Lourdes Patricia Portugal Poma Costa**

**Orientador: Prof. Dr. Cesar Augusto Cavalheiro Marcondes**

São Carlos – SP

Junho/2013

**UNIVERSIDADE FEDERAL DE SÃO CARLOS**

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# APRIMORANDO O DESEMPENHO DE ALGORITMOS DE ROTEAMENTO EM VANETS UTILIZANDO CLASSIFICAÇÃO

**Lourdes Patricia Portugal Poma Costa**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Sistemas Distribuídos e Redes de Computadores
Orientador: Prof. Dr. Cesar Augusto Cavalheiro Marcondes

São Carlos – SP

Junho/2013

**Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar**

C837ad

Costa, Lourdes Patricia Portugal Poma.
Aprimorando o desempenho de algoritmos de roteamento em VANETs utilizando classificação / Lourdes Patricia Portugal Poma Costa. -- São Carlos : UFSCar, 2013.
94 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2013.

1. Redes de computação. 2. Redes tolerantes a atrasos e desconexões. 3. Sistemas de comunicação móvel. 4. Ad hoc networks (Redes de computação). I. Título.

CDD: 004.6 (20ª)

# Universidade Federal de São Carlos

## Centro de Ciências Exatas e de Tecnologia

### Programa de Pós-Graduação em Ciência da Computação

# "Aprimorando o Desempenho de Algoritmos de Roteamento em Vanets utilizando Classificação"

Lourdes Patricia Portugal Poma Costa

**Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação**

**Membros da Banca:**

**Prof. Dr. Cesar Augusto Cavalheiro Marcondes**
**(Orientador - DC/UFSCar)**

**Prof. Dr. Hermes Senger**
**(DC/UFSCar)**

**Profa. Dra. Luciana Arantes**
**(University Pierre et Marie Curie)**

**São Carlos**
**Julho/2013**

À minha família. A Deus.

# Agradecimentos

Ao meu orientador, Prof. Dr. Cesar Marcondes, pela orientação, apoio e a oportunidade de vir do exterior, que fez possível os meus desejos de fazer pesquisa.

À minha mãe e ao meu pai, porque ainda na distância sempre estiveram comigo.

À minha irmã Ana, por a sua amizade e confiança.

Ao meu irmão Carlos, por me incentivar nos meus estudos, sendo o reflexo do profissional que desejo ser.

À minha tia Aurora. Não houve momento de dificuldade que não lembrei da sua valentia.

Ao meu esposo Alceu, pelo amor e apoio.

Ao todo o pessoal do Departamento de Computação da UFSCAR, aos companheiros, professores e amigos.

À CAPES pelo apoio financeiro, que tornou possível este trabalho.

A Deus, por tudo.

# Resumo

As *VANETs* são redes de veículos com capacidade de estabelecer comunicações sem fio entre veículos e com equipamentos nas estradas. Estas redes poderiam ser usadas para a transferência de dados de diversas aplicações. No entanto, devido à mobilidade dos veículos, as *VANETs* apresentam conectividade intermitente entre os nós, dificultando a transmissão de mensagens. Ante a impossibilidade de ter conectividade de fim a fim, as mensagens são encaminhadas progressivamente de veículo em veículo, e armazenadas quando não houver a possibilidade de retransmitir. Adicionalmente, para incrementar a probabilidade de entrega, as mensagens são replicadas e disseminadas pela rede. Não obstante, a replicação de mensagens pode gerar alta sobrecarga de rede e alto consumo de recursos. Por causa disto, projetaram-se algoritmos para cenários específicos de: baixa, moderada e alta conectividade. Estes algoritmos, quando aplicados em ambientes de zonas de diferente densidade de nós,como cidades, podem diminuir o seu desempenho pela falta da capacidade de se adaptar a diferentes condições de conectividade. Contudo, neste trabalho foi desenvolvido um método para adaptar o comportamento dos algoritmos de roteamento por replicação de mensagens a diferentes situações de conectividade segundo a densidade das zonas onde se movimentam os nós retransmissores. O método consiste em três fases. Na primeira, são coletados os dados dos eventos de repasse de mensagens utilizando o algoritmo de roteamento padrão. Na segunda fase, utilizam-se os dados coletados para treinar um classificador baseado em árvores de decisão. Na última fase, o classificador é então empregado para determinar se uma situação de repasse de mensagem é favorável segundo a densidade de nós. Desta forma, os algoritmos de roteamento podem decidir se repassar ou não uma mensagem com o suporte do classificador. Esta abordagem foi avaliada com traces de movimentos reais, para aprimorar o desempenho dos algoritmos de roteamento *Spray and Wait* e *Epidemic*. Os resultados dos experimentos realizados revelam que esta abordagem pode contribuir para o aprimoramento do desempenho.

**Palavras-chave**: Redes tolerantes a atrasos e desconexões, DTNs, Redes Móveis, Redes Móveis Ad-Hoc, MANETs, Redes Veiculares, VANETs

# Abstract

Vehicular ad-hoc networks (*VANETs*) are networks capable of establishing communications between vehicles and road-side units. *VANETs* could be employed in data transmission applications. However, due to vehicle mobility, *VANETs* present intermittent connectivity, making message transmission a challenging task. Due to the lack of an end-to-end connectivity, messages are forwarded from vehicle to vehicle and stored when it is not possible to retransmit. Additionally, in order to improve delivery probability, messages are replicated and disseminated over the network. However, message replication may cause high network overhead and resource usage. As result, considerable research effort has been devoted to develop algorithms for specific scenarios: low, moderate and high connectivity. Nevertheless, algorithms projected for scenarios with a specific connectivity lack the ability to adapt to situations with zones presenting different node density. This lack of adaptation may negatively affect the performance in application such as data transmission in cities. This masters project proposes develops a method to automatically adapt message replication routing algorithms to different node density scenarios. The proposed method is composed of three phases. The first phase collects data from message retransmission events using a standard routing algorithms. The second phase consists in training a decision tree classifier based on the collected data. Finally, in the third phase the trained classifier is used to determine whether a message should be retransmitted or not based on the local node density. Therefore, the proposed method allows routing algorithms to query the trained classifier to decide if a message should be retransmitted. The proposed method was evaluated with real movement traces in order to improve *Spray and Wait* and *Epidemic* routing algorithms. Results indicate that the proposed method may contribute to performance enhancement.

**Keywords**: Delay and Disruption Tolerant Networks, DTNs, Mobile Networks, Mobile Ad-Hoc Networks, MANETs, Vehicular Networks, VANETs

# Lista de Figuras

# Lista de Tabelas

# Sumário

# Capítulo 1

# Introdução

## 1.1 Motivação

As redes móveis veiculares *Ad Hoc* (*Vehicular Ad hoc Networks - VANETs*) são redes móveis de veículos com autonomia para estabelecer comunicação sem fio entre veículos e com unidades de transmissão sem fio nas estradas.

A alta aceitação das aplicações de segurança rodoviária, os avanços na computação e nas tecnologias de comunicação sem fio têm convertido as *VANETs* em um tema muito pesquisado em tópicos como (VASILAKOS; ZHANG; SPYROPOULOS, 2011): tecnologias em nível físico e de camada de enlace, roteamento e disseminação de dados, modelos de mobilidade e simulação e, privacidade e segurança de dados.

Uma vez que os veículos podem ser equipados com computadores sem limitações críticas de energia, os nós das *VANETs* têm uma maior capacidade de armazenamento de dados e processamento quando comparados a outros tipos nós em redes móveis. Por causa disto, espera-se que as *VANETs* possam se aplicar com sucesso a diversos tipos de aplicações, como por exemplo, aplicações de monitoramento nas estradas, monitoramento ambiental e do tráfego veicular (HULL et al., 2006) (DIKAIAKOS et al., 2005). Adicionalmente, o baixo custo operacional das *VANETs*, em relação às redes baseadas em infraestrutura, determina muito interesse na utilização das *VANETs* para transferência de dados de aplicações tolerantes a atraso. Deste modo, as *VANETs* poderiam se utilizar para estender a cobertura de redes baseadas em infraestrutura.

## 1.2 Descrição do problema

Apesar do futuro promissor das *VANET* como redes de suporte ou redes de transmissão, as *VANETs* são redes de topologia altamente dinâmica e de conectividade intermitente. Assim, a viabilidade das aplicações depende de como os algoritmos de roteamento resolvem problemas de desconexão.

Este projeto refere-se às *VANETs* em ambientes urbanos para a transmissão de dados de aplicações tolerantes a atraso. Nestes tipos de aplicações, o problema de conectividade intermitente pode ser resolvido realizando o roteamento progressivamente, usando veículos como retransmissores que repassam a mensagem a outros veículos até a mensagem chegar ao veículo destino. Quando um veículo não detectar outro retransmissor, a mensagem pode ser armazenada até o veículo encontrar outra oportunidade para retransmitir. Este paradigma de roteamento é conhecido como paradigma de armazenamento, transporte e encaminhamento das mensagens (*store-carry-and-forward*) e pode reduzir a perda de mensagens pelas desconexões de rede (FRANCK; GIL-CASTINEIRA, 2007).

Em ambientes urbanos, as *VANETs* apresentam outro desafio adicional ao problema de conectividade intermitente. O desempenho da transmissão de dados está influenciado pela congestão veicular, que por sua vez determina a congestão do meio de transmissão. Desta forma, a existência de zonas de diferentes densidades de veículos em uma cidade, criam diferentes situações e cenários de conectividade. Estes cenários podem incluir condições adversas para a transmissão de mensagens: redes de veículos esparsas e redes de veículos com componentes altamente conectados.

Existem algoritmos de roteamento e de disseminação de dados que foram projetados para agir em cenários adversos de transmissão: redes esparsas ou de pouca densidade (VAHDAT; BECKER, 2000), (HARRAS; ALMEROTH; BELDING-ROYER, 2005), ou redes densas com alta conectividade (SPYROPOULOS; PSOUNIS; RAGHAVENDRA, 2005). Não obstante, em ambientes com uma distribuição de nós no espaço não uniforme, o desempenho destes algoritmos pode ser prejudicado. A presença de zonas de diferente densidade implica a necessidade de mecanismos que permitam adaptar o comportamento dos algoritmos de roteamento a diferentes situações e condições de conectividade de rede.

## 1.3 Hipóteses do projeto

Este projeto de mestrado tem como hipótese que o desempenho dos algoritmos de roteamento que usam o paradigma *store-carry-and-forward*, aplicados ao roteamento de mensagens em *VANETs* urbanas, pode ser aprimorado pelo conhecimento das condições da rede em diversas zonas, determinadas pela densidade dos veículos nos momentos dos repasses das mensagens por cada retransmissor.

Este projeto também considera que veículos com movimento padronizado, como ônibus que seguem rotas e horários de transporte, poderiam usar algoritmos de classificação como suporte para a predição de determinados eventos relacionados aos repasses das mensagens. Deste modo, os classificadores poderiam ser treinados com informação do funcionamento padrão dos algoritmos e, posteriormente, aprimorados alterando a lógica interna dos algoritmos para poder decidir repassar uma mensagem ou não, segundo as predições do classificador treinado.

## 1.4 Principais contribuições deste projeto

A primeira contribuição deste trabalho, é a avaliação dos algoritmos de roteamento em cenários que usam *traces* de movimento real em grande escala (254 veículos em média). Os algoritmos de roteamento avaliados foram: *Epidemic*, *Spray and Wait* e *Prophet*. Estes algoritmos são altamente estudados, porém; em trabalhos como (NELSON; BAKHT; KRAVETS, 2009), a representação do movimento é realizada com movimentos sintéticos que podem diminuir o impacto do comportamento dos algoritmos em diferentes condições de conectividade. Considera-se importante a utilização de *traces* de movimento real, uma vez que a limitação do movimento dos veículos geram zonas de diferente densidade: baixa, moderada ou alta conectividade, que podem influenciar no desempenho dos algoritmos.

A segunda contribuição é a comparação das características de movimento e conectividade dos modelos sintéticos *Random Waitpoint* e *Shortest Path Map Based* com as características encontradas nos *traces* de movimento real. A relação destas características com os resultados de desempenho dos algoritmos de roteamento avaliados, revelam maior impacto no *overhead* de rede nos cenários com *traces* de movimento, quando comparados com os modelos de movimento sintéticos.

A última contribuição deste projeto é a de dar continuidade a uma proposta ainda não muito explorada, porém promissória. Esta proposta consiste na utilização de classifica-

dores no processo de roteamento de mensagens em *VANETs*. Deste modo, classificadores podem ser treinados com tuplas de dados de eventos de repasse de mensagens, entre nós retransmissores. Cada tupla está relacionada com um atributo de anotação de classe que, para o algoritmo de roteamento, significa que tão favorável foi o repasse da mensagem nesse momento. Posteriormente, os nós retransmissores antes de repassar uma mensagem, descrevem a oportunidade de retransmissão através de um conjunto de valores de atributos. Estes valores são incluídos em uma tupla não anotada, a qual é enviada ao classificador treinado para obter o seu atributo de anotação de classe. Segundo o valor do atributo de anotação de classe obtido, o algoritmo de roteamento repassa ou não a mensagem. Esta abordagem aplicou-se aos algoritmos de roteamento *Epidemic* e *Spray and Wait*, com o alvo de que possam reconhecer, com auxilio do classificador, oportunidades de repasse de mensagem favoráveis segundo a densidade de veículos de uma zona.

## 1.5 Organização da monografia

Esta dissertação está organizada da seguinte maneira. No Capítulo (2), são apresentados os principais conceitos das *VANETs*, os principais paradigmas de roteamento aplicados a este tipo de rede, e, são incluídas algumas métricas de caracterização da conectividade e movimento em redes móveis. No Capítulo (3), descrevem-se o processo de simulação, os dados de movimento reais utilizados, os cenários de estudo e, inclui-se uma caracterização da *VANETs* estudada e a sua diferenciação com outras redes criadas por modelos de movimento sintéticos. No Capítulo (4), apresenta-se a abordagem de classificação, descrevendo os atributos escolhidos para a classificação, o método aplicado para realizar roteamento com suporte de um classificador, e mostram-se os resultados de desempenho da aplicação da abordagem proposta aos protocolos de roteamento *SaW* (SPYROPOULOS; PSOUNIS; RAGHAVENDRA, 2005) e *Epidemic*(VAHDAT; BECKER, 2000). No Capítulo (5.0.1), são apresentadas as conclusões e possibilidades de pesquisas futuras decorrentes deste trabalho.

# Capítulo 2

## Roteamento em VANETs

Redes veiculares Ad Hoc (*Vehicular Ad hoc Networks - VANETs*) são redes móveis (*Mobile Ad Hoc Networks - MANETs*) de veículos com autonomia para estabelecer comunicação sem fio entre veículos (*vehicle to vehicle - V2V*) e comunicação com apoio de infraestrutura (*vehicle to infrastructure* ou *infrastructure to vehicle - V2I* ou *I2V*).

Por serem *MANETs*, as *VANETs* apresentam as seguintes características: são redes de topologia dinâmica, executam algoritmos de roteamento em ambientes distribuídos e estão formadas por nós móveis que podem ser clientes ou servidores de dados. Estas duas últimas características se assemelham às redes *P2P* e, existem diferentes abordagens que aproveitam essas similaridades para adotar funcionalidades das *P2P* em processos como descoberta e disseminação de conteúdo (LEE et al., 2006), (ZHANG; ZHAO; CAO, 2009), (LI; YANG; LOU, 2011).

As *VANETs*, no entanto, apresentam características adicionais que reduzem a efetividade dos algoritmos de roteamento tradicionais projetados para outros tipos de *MANETs*. As mais relevantes destas características são:

**Topologia altamente dinâmica e conexões intermitentes:** Diferentemente de outros tipos de *MANETs*, os nós das *VANETs* podem alcançar velocidades altas alterando rapidamente a topologia da rede. Adicionalmente, o impacto ambiental sobre a propagação dos sinais de rádio, criam problemas de reflexão e atenuação que ocasionam condições adversas de transmissão (HARTENSTEIN; LABERTEAUX, 2008). Deste modo, podem existir nós isolados ou componentes de rede com conectividade intermitente entre eles.

**Movimento delimitado:** O movimento dos veículos é limitado segundo a estrutura do ambiente onde os nós se deslocam (urbano ou rural) e a fatores como: congestão

do tráfego veicular, estabelecimento de rotas de trânsito, sinais de tráfego veicular. No caso dos ônibus, apresentam-se padrões de movimento mais característicos como rotas e horários. Tanto assim, que existem estudos de *traces* de movimento de ônibus, cujo objetivo é obter modelos matemáticos para a geração de traces sintéticos que possam ser utilizados na simulação de diferentes cenários (ZHANG et al., 2007), (DOERING et al., 2010).

**Densidade variável:** A distribuição dos nós no ambiente não é uniforme e cria zonas de diferentes densidades.Por exemplo, nos cruzamentos entre ruas em ambientes urbanos.

**Recursos não limitados de energia e armazenamento:** Os veículos não têm limitações críticas de energia. Assim, podem ser equipados com recursos de maiores capacidades de armazenamento e processamento, quando comparados com outros tipos de nós de redes móveis. Não obstante, o meio de transmissão compartilhado é um recurso limitado e, uma das dificuldades da transmissão de mensagens em redes móveis, é o uso eficiente deste recurso.

Estas características particulares das *VANETs* devem ser consideradas no projeto de algoritmos de roteamento. Nas seções que seguem serão apresentadas as principais abordagens de roteamento em *VANETs*, destacando a comparação entre elas e o seu relacionamento no desempenho. Apresenta-se também, conceitos de teoria de grafos e classificadores que podem ser aplicados como ferramentas no projeto de algoritmos de roteamento para *VANETs*.

## 2.1 Paradigmas de roteamento de mensagens

*VANETs* podem empregar paradigmas de roteamento não tradicionais. Assim, nesta dissertação, o termo roteamento refere-se ao envio de uma mensagem de um nó origem a um nó destino, sem implicar necessariamente a utilização de tabelas de rotas. Precisa-se desta especificação para o termo roteamento, pois parte dos paradigmas não tradicionais prescindem de tabelas de rotas.

Entre os paradigmas de roteamento para *VANETs* mais referenciados temos: (i) paradigma de roteamento baseado em topologia, (ii) paradigma de roteamento baseado em informação geográfica e, (iii) paradigma de roteamento baseado em armazenamento de mensagem. O paradigma de maior importância para este trabalho de pesquisa é o para-

digma (iii). Não obstante, os paradigmas (i) e (ii) são descritos nesta dissertação para compará-los com o paradigma (iii).

A escolha de um paradigma depende da aplicação particular das *VANETs*. Por exemplo, aplicações de segurança rodoviária não são tolerantes ao atraso e são propensas a problemas de congestão. Isto ocorre porque as transmissões são feitas em *broadcast*/*multicast*(ZEADALLY et al., 2012) para informar de algum evento a vários nós dentro de uma determinada área ou localização (*geocasting*). Ademais, as mensagens deste tipo de aplicações deveriam ter um atraso máximo de 100 $\mu$ s (KARAGIANNIS et al., 2011).Por esse motivo, o paradigma (iii), de armazenamento de mensagens, não é apropriado para este tipo de aplicação. No entanto, existem aplicações para as quais o paradigma de armazenamento de mensagens se mostra o mais adequado para realizar o roteamento.

### 2.1.1 Paradigma de roteamento baseado em topologia

O paradigma de roteamento baseado em topologia utiliza tabelas de rotas para encaminhar mensagens e é conhecido como o paradigma tradicional das *MANETs*. A existência de uma rota é determinada pelo nó origem antes de iniciar a transmissão de uma mensagem ao nó destino (roteamento baseado na origem). Cada rota consiste em uma série de identificadores de nós retransmissores.

Existem dois mecanismos para construir uma tabela de rotas. O primeiro, utilizado pelos algoritmos de roteamento proativos, consiste na manutenção periódica da toda a tabela de rotas. O segundo, utilizado pelos algoritmos reativos, consiste no processo de descobrimento de rota, previamente ao envio de uma mensagem entre a origem e o destino.

Uma vez que foram originalmente propostos para *MANETs*, o paradigma baseado em topologia apresenta limitações quando aplicado às VANETs. O paradigma baseado em topologia não é adequado para encontrar rotas fim a fim em cenários esparsos, de pouca densidade de nós ou de conectividade intermitente entre nós. Outro problema é a excessiva sobrecarga de rede que pode causar pelo descobrimento e a manutenção de rotas em uma topologia altamente dinâmica como as *VANETs* (KARAGIANNIS et al., 2011). Este problema é agravado em protocolos proativos que reconstroem toda a tabela de rotas periodicamente. Os algoritmos de roteamento proativos geram considerável sobrecarga de rede, em redes a extensas e com rápido movimento de nós (BASAGNI; CONTI; GIORDANO, 2004).

Embora o paradigma baseado na topologia tenha limitações, utiliza-se em aplicações

de segurança rodoviária devido ao requerimento de baixo atraso de envio. Uma abordagem para melhorar a estabilidade dos enlaces e reduzir a sobrecarga de rede, é utilizar informação da mobilidade dos nós para predizer futuras desconexões. Desta maneira, os algoritmos poderiam determinar quanto tempo durará uma rota, e iniciar o procura de uma nova antes de sofrer algum problema de desconexão(KARAGIANNIS et al., 2011).

### 2.1.2 Paradigma baseado em informação geográfica

Neste paradigma a seleção dos nós intermediários depende da localização geográfica do nó emissor, dos nós vizinhos e do nó destino. Realiza-se um processo de roteamento incremental, diferentemente do roteamento baseado na origem do paradigma baseado em topologia. Neste processo cada nó escolhe com informação local o próximo nó retransmissor. Precisa-se de um sistema de localização, do intercambio de informação de localização entre nós vizinhos (em *beacons* transmitidos periodicamente); e de um procedimento para encaminhar a mensagem (*broadcast* ou *unicast*)(ZEADALLY et al., 2012).

Uma desvantagem deste paradigma é que o caminho de nós retransmissores usado até o destino não é necessariamente ótimo. Isto se deve a que cada nó decide em base a informação local para escolher um próximo retransmissor a quem repassar a mensagem. Na abordagem mais simples os nós decidem segundo um algoritmo "guloso" ou *greedy routing*. No *greedy routing* um nó escolhe o seguinte retransmissor, de entre seus nós vizinhos, a aquele mais próximo ao nó destino segundo a informação de localização. Desta maneira se assume que cada repasse da mensagem é um avanço na aproximação ao nó destino. Um problema ocasionado pelo *greedy routing* é que os nós poderiam não encontrar um seguinte nó retransmissor próximo ao destino, ainda existindo alguma rota de nós intermediários. Na Figura (2.1), onde as linhas retas indicam conectividade entre os nós, é mostrado um exemplo citado em (KARP; KUNG, 2000). Neste exemplo, uma mensagem dirigida ao nó $D$ foi encaminha desde o nó $y$ ao nó $x$ usando *greedy routing*. Posteriormente, quando $x$ procurar outro nó mais próximo do que ele ao nó $D$, não encontrará nenhum outro possível retransmissor. Como o nó $x$ não tem conectividade com o nó destino $D$, esta tentativa não será bem sucedida inclusive existindo dois caminhos alternativos para chegar ao nó destino. Para tratar este tipo de problemas alguns algoritmos consideram procedimentos adicionais. No entanto, estes procedimentos estão próximos a agir como os algoritmos baseados na topologia, e consomem recursos de rede na procura de soluções alternativas. Por exemplo, o algoritmo do protocolo *GPSR* (*Greedy Perimeter Stateless Routing*)(KARP; KUNG, 2000), confere se existem rotas alternativas até o destino para

executar depois um algoritmo denominado "*the right-hand rule*" e, deste modo, determinar um nó retransmissor. Outro método usado no protocolo *GSR* (*Geographic Source Routing*)(LOCHERT et al., 2003), é similar ao modo de proceder dos protocolos reativos baseados em topologia. *GSR* inunda a rede com mensagens de requisição solicitando a localização do nó destino. Quando a mensagem de requisição alcançar o nó destino, ele responde com a sua localização e o nó emissor executa o algoritmo *Dijkstra shortest path* utilizando informação de um mapa para encontrar um caminho até o destino.

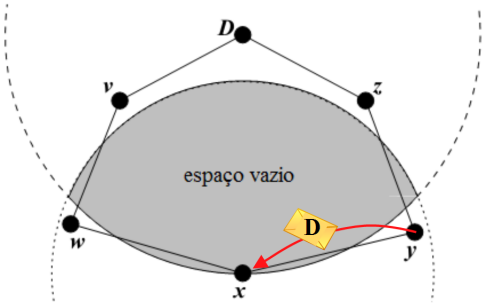



**Figura 2.1: Falha do *greedy routing*.**

Os algoritmos de roteamento baseados em informação geográfica são utilizados usualmente para a disseminação de mensagens dentro de uma área geográfica determinada (*geocasting*) (CASTEIGTS; NAYAK; STOJMENOVIC, 2011). Não obstante, este tipo de paradigma também pode se utilizar para transmissão de mensagens *unicast*. Pelo escopo do tema desta dissertação, não se inclui informação de algoritmos próprios deste paradigma, porém, informação adicional de algoritmos de roteamento *unicast* que utilizam informação geográfica pode se encontrar em (KARAGIANNIS et al., 2011) e (BERNSEN; MANIVANNAN, 2009).

Os protocolos deste paradigma estão muito associados às *VANETs*, pela suposição de que os veículos são nós que facilmente poderiam contar com algum sistema de localização geográfica. Contudo, as características das *VANETs* e os obstáculos no ambiente em que agem não permitem passar por alto o problema das conexões intermitentes ou descontinuas. Por este motivo alguns algoritmos híbridos, baseados neste paradigma, consideram o paradigma de armazenamento de mensagens, que é o próximo e último paradigma que se considera nesta dissertação. Em razão disto, podemos agrupar os algoritmos de roteamento baseados neste paradigma como: (i) baseados totalmente em informação geográfica (*Non-DTN*) e, (ii) baseados em *DTNs* que utilizam o paradigma de armazenamento de mensagens (CHENG et al., 2010).

### 2.1.3 Paradigma de roteamento baseado em armazenamento de mensagens

*VANETs* apresentam desafios no roteamento de mensagens por causa da alta mobilidade dos veículos. Para aplicações tolerantes a atraso uma solução é utilizar o paradigma de roteamento baseado em armazenamento, transporte e encaminhamento das mensagens (*store-carry-and-forward*), paradigma referido nesta dissertação como paradigma baseado em armazenamento de mensagens.

O paradigma baseado em armazenamento de mensagens é próprio das redes tolerantes a atrasos e desconexões (*Delay and Disruption Tolerant Networks - DTNs*) (VASILAKOS; ZHANG; SPYROPOULOS, 2011) e pode ser aplicado a *VANETs* para aprimorar o desempenho em cenários esparsos, ou de frequentes desconexões (FRANCK; GIL-CASTINEIRA, 2007).

Este paradigma não usa tabelas de rotas para encaminhar as mensagens e, de maneira similar com o paradigma de roteamento baseado em informação geográfica, realiza um encaminhamento progressivo (*hop -by -hop*). No processo de encaminhamento, cada nó repassa a mensagem a nós intermediários dentro do raio de alcance, sucessivamente até a mensagem chegar ao nó destino. Quando não houver um nó vizinho que possa ser usado como retransmissor, a mensagem é armazenada até uma nova oportunidade de retransmissão.

Em razão de que os nós não se utilizam tabelas de rotas, os nós usam duas estratégias para encaminhar as mensagens ao destino: (i) roteamento de uma cópia e, (ii) roteamento de múltiplas cópias. As duas estratégias são apresentadas a seguir.

**Algoritmos de roteamento de uma cópia (*single-copy*):** Também chamados de algoritmos de roteamento por encaminhamento ou *forwarding*. Consiste em manter uma só cópia da mensagem em toda a rede. No encaminhamento, cada nó repassa a mensagem a um nó intermediário e a exclui da memória, sucessivamente, até a mensagem chegar ao destino. Desta maneira só uma cópia da mensagem é mantida na rede e se consumem baixos recursos de armazenamento, energia e uso do meio de transmissão.

**Algoritmos de roteamento de múltiplas cópias (*multi-copy*):** Também conhecidos como algoritmos de roteamento por replicação de mensagens. Estes algoritmos repassam réplicas ou cópias de uma mensagem em cada transmissão a um nó intermediário. Este procedimento repete-se sucessivamente para disseminar cópias pela

rede e assim aumentar a probabilidade da entrega e reduzir o atraso de envio.

Existem alguns problemas associados pela replicação das mensagens: o alto consumo de recursos e a excessiva disseminação de cópias que, sem medidas de controle, pode inundar a rede (*flooding*). A disseminação de copias quando não for controlada, é um processo similar à disseminação de uma epidemia. Deste modo, um dos exemplos mais representativos deste tipo de algoritmos de roteamento é o algoritmo de roteamento *Epidemic* (VAHDAT; BECKER, 2000).

Embora os algoritmos deste tipo possam obter bom desempenho em redes esparsas e de pouca densidade, a disseminação de cópias sem controle pode causar que o desempenho do sistema seja baixo em cenários de mediana a alta densidade. Em redes de grande escala leva a um consumo de recursos extenso causando significativas sobrecargas de rede. A pesar do alto consumo de recursos, neste projeto de mestrado escolheram-se os algoritmos de roteamento por replicação de mensagens para o seu estudo e posterior aprimoramento. Isto é porque se consideram uma boa alternativa em *VANETs* por serem redes sem limitações críticas de espaço de armazenamento e energia. O desafio está principalmente em projetar algoritmos que diminuam a sobrecarga do meio de transmissão limitado

Para discernir entre o paradigma de roteamento baseado em armazenamento das mensagens e o paradigma de roteamento tradicional das *MANETs*, apresenta-se a continuação a Tabela (2.1).

**Tabela 2.1: Comparação entre o paradigma baseado em topologia e o paradigma baseado em armazenamento das mensagens.**

| | Baseado em topologia | Baseado em armazenamento |
|---|---|---|
| Conectividade | Não enviam a mensagem até encontrar uma rota até o destino. | Quando não houver retransmissores armazenam a mensagem e retransmitem em outra oportunidade. |
| Armazenamento | Armazena as mensagens por pouco tempo (*buffering* de ordem de $\mu s$). | Armazena as mensagens por tempo prolongado (minutos, horas). |
| Mobilidade | A alta mobilidade prejudica a retransmissão pela dificuldade na manutenção das tabelas de rotas. | A mobilidade é aproveitada para disseminar ou retransmitir a mensagem em novos contatos que o nó possa estabelecer. |
| Atraso | Entrega com menor atraso, porém, a probabilidade de entrega pode diminuir em cenários esparsos. | Maior atraso pelo armazenamento de mensagens dependendo do tempo entre contatos. |

**Efeitos do armazenamento de mensagens:** Em (FRANCK; GIL-CASTINEIRA, 2007) se avaliou o desempenho em termos de atraso e taxa de perda de pacotes, do paradigma de roteamento baseado em topologia das *MANETs*, contra o paradigma de armazenamento de mensagens. O experimento realizou-se em um cenário simulado de uma rede formada por 10 veículos, distribuídos aleatoriamente em uma área de $2km^2$ e, seguindo um modelo de movimento sintético. O encaminhamento das mensagens realizou-se em *broadcasts* retransmitidos cada vez que um nó recebe uma mensagem. Se um nó não tivesse nós vizinhos na tentativa de realizar o *broadcast*, a mensagem é excluída no caso do paradigma tradicional. No caso do paradigma de armazenamento, a mensagem será armazenada para ser retransmitida em outra oportunidade.

Na Figura (2.2) é mostrado o comparativo em perda de pacotes das duas abordagens segundo (FRANCK; GIL-CASTINEIRA, 2007). Em ambos os casos, a maior número de nós vizinhos, menores são as perdas de pacotes. A abordagem tradicional apresenta maior quantidade em perdas em todos os casos do que a abordagem de armazenamento de mensagens. Observou-se que em cenários de pouca a mediana densidade, a mensagem tem a probabilidade de mais do 50% de não ser retransmitida no paradigma tradicional.

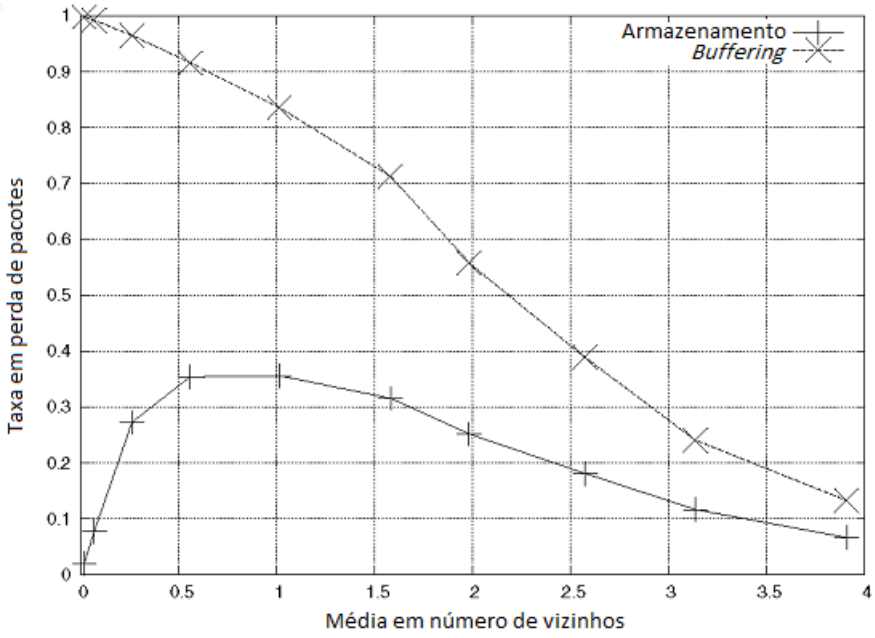



**Figura 2.2: Comparativo da taxa de perda de pacotes no paradigma baseado em topologia e o paradigma de armazenamento.**

Na Tabela (2.2) é mostrado o resultado da avaliação do atraso de entrega em am-

bos os paradigmas segundo (FRANCK; GIL-CASTINEIRA, 2007). No paradigma de armazenamento de mensagens considera-se, adicionalmente ao atraso de envio pela transmissão, o tempo que as mensagens ficaram armazenadas até que os nós portadores estabelecerem contacto com outros nós que não possuíssem a mensagem. No caso do paradigma tradicional, considere-se que a transmissão de um pacote de um nó a outro consume aproximadamente uma UT(unidade de tempo). Deste modo, o atraso de fim a fim em UTs será igual ao número de *hops*. Na Tabela (2.2) mostra-se os resultados do tempo que demoraram as mensagem em chegar até o destino em cenários: (i)esparsos, (ii)de densidade moderada e, (iii) densos.

No paradigma de armazenamento, o atraso é crítico em cenários esparsos. Conforme a densidade aumenta observa-se dois comportamentos distintos nestas duas abordagens. No paradigma de armazenamento, o atraso diminui porque a probabilidade de estabelecer contato com um nó, é maior. No caso do paradigma tradicional o atraso aumenta. Isto acontece porque as mensagens podem ser retransmitidas a mais quantidade de saltos e não são excluídas rapidamente. Assim, o tempo que permanecem na rede é maior.

**Tabela 2.2: Comparação do atraso de fim a fim, em UTs, segundo diferentes densidades de rede.**

| Cenário | | Mín. | Máx. | Média | Desv. Pad. | Perc. 0.9 |
|---|---|---|---|---|---|---|
| *B. Na topologia* | Esparsa | 1.0 | 3.0 | 1.029 | 0.2 | 1.0 |
| | Moderada | 1.0 | 12.0 | 1.43 | 0.75 | 2.0 |
| | Densa | 1.0 | 13.0 | 1.83 | 0.98 | 3.0 |
| Armazenamento | Esparsa | 1.0 | 32681.0 | 3218.13 | 3154.04 | 7391.0 |
| | Moderada | 1.0 | 3051.0 | 257.27 | 306.72 | 671.2 |
| | Densa | 1.0 | 702.0 | 16.72 | 47.85 | 52.0 |

Os resultados encontrados em (FRANCK; GIL-CASTINEIRA, 2007) são um exemplo claro da relação entre perda de pacotes e atraso nos dos paradigmas avaliados. Assim, existe um sacrifício do atraso de envio para diminuir a perda de pacotes e, em alguns casos, aumentar a probabilidade de entrega, quando se utiliza o paradigma de armazenamento de mensagens.

Resumindo, nesta seção descreveram-se os principais paradigmas de roteamento em *VANETs*. O paradigma baseado na topologia tem a vantagem de poder encontrar rotas ótimas, uma vez que, se conhece a topologia da rede. Não obstante, está abordagem pode não ser escalável pela dificuldade da manutenção de rotas quando a densidade de nós for

elevada (especialmente em protocolos proativos). O paradigma baseado na localização resolve o problema de baixa escalabilidade, liberando aos nós da dificuldade de manter um caminho fixo de nós intermediários até chegar ao destino. No entanto, o paradigma baseado na localização apresenta a desvantagem de que os caminhos utilizados para o envio, podem não ser os melhores por usar informação local e limitada da topologia. Ambos os paradigmas são pouco viáveis em redes esparsas e em cenários de conexões intermitentes entre nós. Neste tipo de cenários, o paradigma de armazenamento de mensagens pode diminuir a perda de pacotes e incrementar a probabilidade de entrega. Porém, incrementado o atraso de entrega.

## 2.2 Classificação de algoritmos de roteamento por replicação de mensagem ou de múltiplas cópias

Estes algoritmos de roteamento repassam cópias de uma mensagem, disseminando-as pela rede através de contatos entre nós (ver Seção (2.1.3)).

Em *VANETs*, estes algoritmos podem ser utilizados porque os veículos não tem limitações críticas de recursos de energia e espaço de armazenamento. Não obstante, precisa-se resolver o problema de sobrecarga e congestão do meio de transmissão. As soluções utilizadas são basicamente duas: (i) reduzir o número de retransmissões de cópias e, (ii) excluir cópias disseminadas pela rede, para evitar a inundação de cópias. Segundo as estratégias utilizadas para a propagação de cópias pelo meio de transmissão, os algoritmos de roteamento podem se classificar de diferentes formas. Com base na taxonomia de (SPYROPOULOS et al., 2010), mostra-se a seguinte classificação de algoritmos de roteamento *unicast* por replicação de mensagens, em redes de conexões intermitentes. Adicionalmente estudam-se a aplicação de classificadores como suporte na disseminação e roteamento de mensagens.

### 2.2.1 Algoritmos de roteamento por inundação controlada

Estes algoritmos utilizam algum mecanismo para limitar a propagação de cópias por toda a rede. As ações básicas contra a propagação podem ser as seguintes:

***TTL* (*time to live*):** Similar ao valores de *TTL* no *IP*. Quando uma mensagem é configurada com um intervalo de tempo de *TTL*, uma vez transcorrido esse intervalo de tempo, a mensagem é excluída.

**Replicação limitada em tempo:** É um tempo configurado, durante o qual uma mensagem pode ser replicada. Passado este intervalo de tempo, já não se transmitem cópias de uma mensagem.

**Valores de saltos (*hops*):** Inicialmente configura-se um número máximo de saltos ou repasses que se pode realizar de uma mensagem. Em cada repasse, o valor diminui em uma unidade. Se o valor ficar em um, os nós só transmitem diretamente ao nó destino (*direct transmission*).

**Espaço de armazenamento:** Consiste em restringir uma quantidade do espaço de armazenamento para as cópias das mensagens. Quando ultrapassar o limite, os nós excluem geralmente as mensagens mais antigas ou executam um algoritmo para o reemprazo de mensagens.

**Utilização de vacinas:** Inspirado na propagação de epidemias, em (HAAS; SMALL, 2006), estuda-se formas de controle por imunização: (i)*IMMUNE*, (ii)*IMMUNE_TX* e (iii)*VACCINE*. Em (i), as cópias das mensagens excluem-se depois de serem transmitidas com sucesso e os identificadores das mensagens se conservam para criar pacotes de imunidade (*anti-packets*). Estes pacotes evitam futuras reinfecções, ou seja, futuras recepciones de mensagens que já se transmitiram com sucesso. Em (ii), o procedimento é similar ao anterior e, adicionalmente, os pacotes de imunidade são transmitidos a outros nós infectados. Por último, a abordagem (iii) é similar a (ii) e, adicionalmente, os *anti-packets* são retransmitidos também a nós não infectados.

**Contagem do número de cópias:** Consiste em criar um número limitado de cópias de uma mensagem para disseminá-las pela rede. O protocolo *Spray and Wait* (SPYROPOULOS; PSOUNIS; RAGHAVENDRA, 2005) é um dos exemplos mais conhecidos. As pretensões deste tipo de controle são, segundo (SPYROPOULOS; PSOUNIS; RAGHAVENDRA, 2005):

- Gerar um menor número de retransmissões que os algoritmos baseados em inundação de cópias sem controle, em todo tipo de cenário.
- Diminuir a contenção do meio, especialmente em cenários de elevada carga de tráfego de rede.
- Obter valores competitivos de atraso e probabilidade de entrega independentemente da densidade de nós e tamanho da rede.
- Ser viável, de implementação simples e não precisar de informação da rede para operar.

**Controlada segundo probabilidades:** Consiste em retransmitir mensagens com uma probabilidade $p_i$, que poderia ser fixa (previamente configurada) ou calculada em tempo de execução segundo as condições da rede. Baixo este controle cumpre-se que a menor valor de $p_i$, maior o atraso de entrega. A maior valor de $p_i$, maior aproximação à inundação de mensagens (*flooding*).

Segundo (SPYROPOULOS et al., 2010) os protocolos por replicação controlada funcionam bem em cenários de nós homogêneos: nós de patrões de movimento e transmissão de dados similares, e que transitam frequentemente pela rede. Em outro tipo de cenários, pode ser necessário coletar informação estadística ou de rede para decisões mais inteligentes de como levar a mensagem ao nó destino.

A continuação explica-se o funcionamento do algoritmo *Epidemic*, que é o exemplo mais representativo do roteamento baseado em replicação de mensagens. Este algoritmo é incluído nesta seção, porque embora na sua implementação mais simples realize encaminhamento de mensagens por inundação de cópias de mensagens, em outras implementações pode utilizar medidas básicas de controle. Posteriormente inclui-se também ao algoritmo *Spray and Wait*, que é um algoritmo de roteamento de controle de inundação por contagem do número de cópias

#### 2.2.1.1 Algoritmo de roteamento *Epidemic*

O algoritmo *Epidemic* é um algoritmo de roteamento por replicação de mensagens para *DTNs*. O envio de mensagens realiza-se repassando cópias das mensagens em oportunidades de contato com os nós vizinhos, de maneira similar à disseminação de uma epidemia. Na sua implementação mais simples, sem mecanismos de controle, o *Epidemic* é um exemplo de encaminhamento de mensagens por inundação de cópias. Na Figura (2.3), onde o eixo horizontal indica os instantes de tempo, mostra-se um exemplo básico de encaminhamento por inundação de cópias. Neste exemplo, o nó $A$ deseja enviar uma mensagem ao nó $G$. No instante $t_1$, o nó origem $A$ repassa uma cópia da mensagem para cada nó vizinho ($C$ e $B$). No instante $t_2$, as cópias da mensagem se propagam pelos nós intermediários. Finalmente no instante $t_3$, uma cópia da mensagem chega até o nó de destino $G$ depois de difundir-se várias cópias da mensagem por toda a rede.

O protocolo *Epidemic* de (VAHDAT; BECKER, 2000), utiliza o algoritmo com algumas variações. As mensagens que um nó possui são indexadas usando uma tabela *hash*. Adicionalmente, utiliza-se um vetor de resumo que contém os valores das chaves ou indexes, da

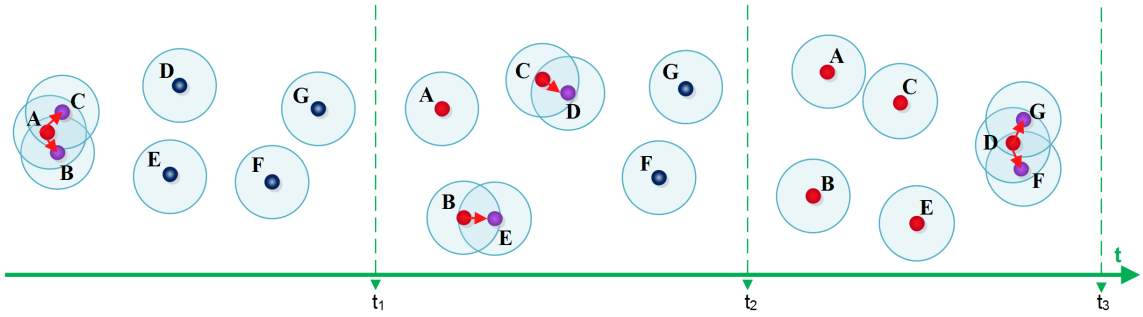



**Figura 2.3: Exemplo da transmissão de mensagem por replicação não controlada.**

tabela *hash*, que referenciam cada mensagem. Quando dois nós estabelecem contato, um dos nós inicia um processo conhecido como *anti-entropy session*. Este processo consiste em enviar o vetor resumo ao outro nó, para que o outro nó possa compará-lo com o vetor resumo dele e, desta maneira, determinar que mensagens precisa solicitar. Na Figura (2.4) é mostrado como se realiza o processo de *anti-entropy session*. Neste exemplo, o nó $A$ informa a $B$ das mensagens que possui enviando o vetor de resumo. Quando o nó $B$ recebe o vetor de resumo de $A$, realiza uma operação lógica $AND$ entre o vetor de $A$ e o dele. Desta forma, o nó $B$ pode determinar que mensagens faltantes poderia obter de $A$. Posteriormente, o nó $B$ solicita as mensagens faltantes ao nó $A$ e o nó $A$ as envia.

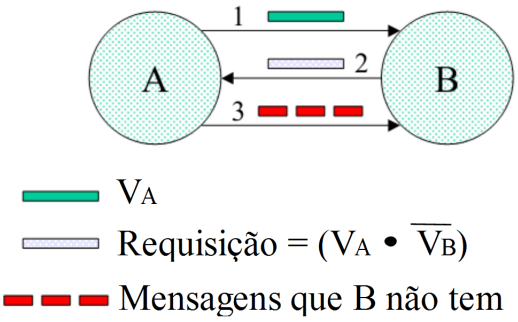



**Figura 2.4: Processo de *anti-entropy* no protocolo *Epidemic* .**

O algoritmo *Epidemic*, pode incluir medidas básicas de controle contra a inundação de cópias. Estas medidas podem ser: valores $TTL$, número máximo de saltos, controle por espaço de armazenamento e imunização por mensagens. Quando comparado com outros algoritmos de roteamento de múltiplas cópias e, que não utilizem informação de rede para operar; em cenários sem limitações drásticas de recursos o algoritmo *Epidemic* pode conseguir menor atraso e maior probabilidade de entrega. Porém em redes de densidade moderada ou alta, gera alta sobrecarga de rede pela retransmissão de cópias. Este

algoritmo é mais adequado para redes esparsas ou de baixa densidade.

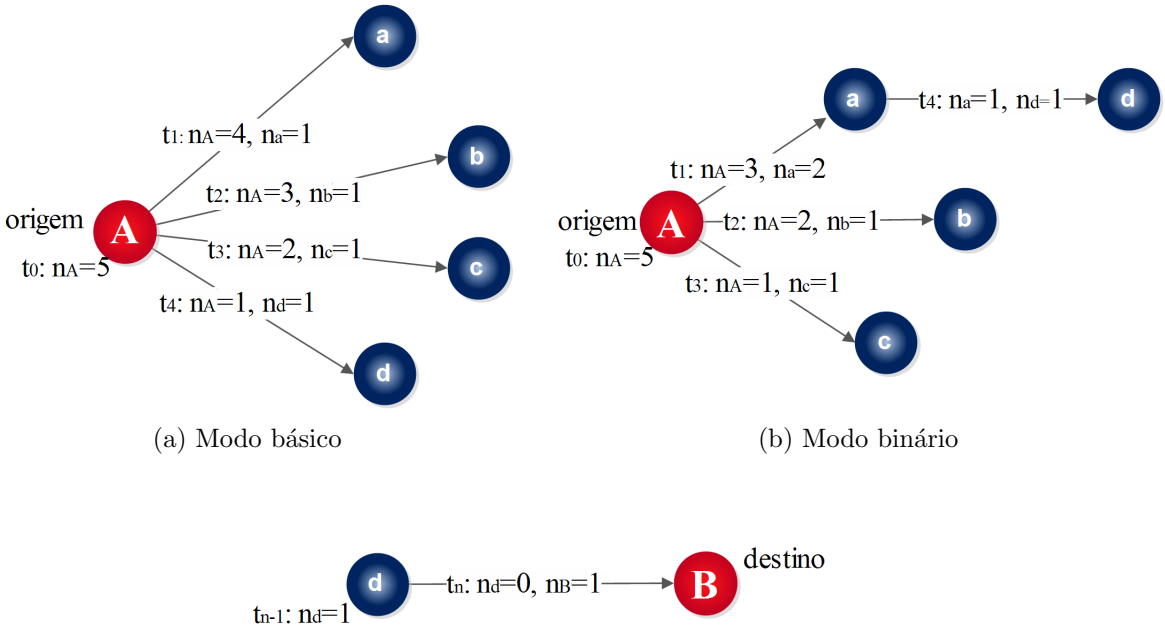
#### 2.2.1.2 Algoritmo de roteamento *Spray and Wait - SaW*

O algoritmo *Spray and Wait* de (SPYROPOULOS; PSOUNIS; RAGHAVENDRA, 2005) é um algoritmo de roteamento por replicação de mensagens com controle de inundação por contagem do número de cópias ($L$). Este algoritmo mantém no máximo $L$ cópias de uma mensagem na rede. A principal vantagem do uso deste algoritmo, é que diminui o número de retransmissões e redundância de dados sem precisar de informação do estado da rede ou dos nós. O algoritmo pode funcionar em um dos seguintes modos:

**Modo básico ou baseado no nó origem:** No modo básico, o nó origem é o único que propaga as cópias da mensagem criada. Inicialmente a mensagem possui um valor de contagem de cópias igual a $L$, onde $L$ é um valor configurado. Cada vez que o nó origem repassa uma cópia a outro nó, o valor de $L$ diminui em uma unidade. Deste modo, o nó origem entrega uma cópia a cada um dos $L-1$ primeiros nós que encontre e que não possuam a mensagem e, reserva uma cópia para si mesmo. Se durante a propagação de cópias não se encontrou ao destino, todo nó que possua uma cópia da mensagem, só a transmite diretamente ao destino (*direct transmission*).

**Modo binário:** Neste modo qualquer nó, origem ou retransmissor, que possua mais de uma cópia ($n$ cópias), pode disseminar cópias de uma mensagem. A disseminação de cópias realiza-se da seguinte maneira. Um nó que possua uma mensagem com contagem de cópias igual a $n$, repassa uma cópia a outro nó com um valor de contagem de cópias de $\lfloor \frac{n}{2} \rfloor$ e, atualiza o valor de $n$ a $\lceil \frac{n}{2} \rceil$ cópias restantes. No principio a propagação de cópias é iniciada pelo nó origem com $n = L$ cópias, onde $L$ é um valor configurado. Se durante a propagação de cópias não se encontrou ao destino, todo nó que possua uma cópia da mensagem, só a transmite diretamente ao destino (*direct transmission*).

Na Figura (2.5) são mostrados os funcionamentos dos dois modos do algoritmo *SaW*. O procedimento começa no tempo $t_0$, quando o nó origem $A$ inicia a propagação de cópias com um valor de $L = 5$. Nos grafos mostrados, cada enlace do tipo $t_i : n_X = x, n_Y = y$ indica o tempo $t_i$ em que aconteceu a transmissão do nó $X$ ao nó $Y$ e o número de contagem de copias que tem cada nó depois da transmissão, ou seja, o número de cópias restantes $x$ no nó $X$ e o número de cópias igual a $y$ do nó $Y$.




(a) Modo básico

(b) Modo binário

(c) Transmissão direta

**Figura 2.5: Exemplo do funcionamento do algoritmo *SaW* com um valor de $L = 5$: (a) Modo básico, (b) modo binário e, (c) em ambos os modos quando o número de cópias de um nó diminui a um, só é permitido transmitir ao nó destino.**

O desempenho deste protocolo depende muito do movimento dos nós, de se eles se deslocam pela área o suficiente como para poder visitar ao nó vizinho. A principal desvantagem na configuração do algoritmo é como determinar que valor de $L$. Embora em (SPYROPOULOS; PSOUNIS; RAGHAVENDRA, 2005) os autores propõem formas de estimar o melhor valor de $L$ para modelos de movimento aleatórios; para diferentes modelos de movimento e condições de rede, é necessário afinar o ajustar o valor de $L$.

### 2.2.2 Algoritmos de roteamento por replicação baseada em utilidade

Nestes algoritmos o repasse de uma cópia a um nó intermediário depende da estimação da qualidade ou utilidade que esse nó represente nessa retransmissão. Para a avaliação ou o cálculo da utilidade é comum usar informação coletada no momento ou estadisticamente, como por exemplo, utilizar informação de contatos anteriores para estimar a utilidade de um nó segundo a sua probabilidade de encontrar-se com o nó destino. Segundo a perspectiva com a qual se determina a utilidade de um nó, as funções de utilidades podem estar classificadas como: (i)dependentes do destino e (ii)independentes do destino.

**Funções dependentes do destino:** Utiliza-se informação da relação dos nós da rede

com o nó destino. Por exemplo, informação histórica baseada em encontros anteriores. O histórico dos encontros passados com o nó destino permite extrair métricas como: frequência de encontros, média de tempo entre encontros, média da duração dos encontros, etc. Assumindo que estes eventos continuarão com a mesma tendência, estas métricas podem se usar para predizer futuros encontros, e distinguir os nós que têm a maior probabilidade de encontrar-se com o destino.

Uma desvantagem neste tipo de protocolos é que o número de nós compromete a escalabilidade desta solução já que se necessita manter atualizados os valores de utilidade para todos nós da rede (LINDGREN; DORIA; SCHELÉN, 2003).

**Funções independentes do destino:** A qualificação dos nós é independiente da relação deles com o nó destino. Assim, um nó poderia ser um bom retransmissor para vários destinos pelas características e capacidades dele. Algumas das características consideradas para avaliação são (SPYROPOULOS et al., 2010):

**Características de mobilidade:** Nós com alta mobilidade tendem a deslocar-se mais no espaço físico e visitar mais nós.

**Recursos do nó:** Nós com boa quantidade de energia e espaço de armazenamento são bons retransmissores. Especialmente em redes heterogêneas.

#### 2.2.2.1 Algoritmo de roteamento Prophet

O algoritmo *Prophet*, (*Probabilistic ROuting Protocol using History of Encounters and Transitivity*)(LINDGREN; DORIA; SCHELÉN, 2003), é um algoritmo de roteamento por replicação de mensagem e baseado em utilidade dependente do nó destino. O cálculo do valor de utilidade dos nós realiza-se da seguinte maneira. Dado um nó qualquer $a$, e um nó destino $b$, o valor de utilidade de $a$ como retransmissor de uma mensagem dirigida a $b$, é calculado segundo a probabilidade de entrega do nó $a$ ao nó $b$, $P_{(a,b)} \in [0,1]$. O valor de $P_{(a,b)}$, depende dos encontros passados entre o nó $a$ e o nó destino $b$. Assim, o protocolo assume que se o nó $a$ encontra frequentemente ao nó $b$, então, o nó $a$ pode ser um bom retransmissor para mensagens dirigidos ao nó $b$.

O funcionamento do protocolo *prophet* pode se resumir em quatro tópicos principais:

**Aumento de $P_{(a,b)}$:** Para assegurar que dois nós que se encontram frequentemente tenham métricas altas que indiquem maior probabilidade de entrega entre eles, o valor da métrica se atualiza cada vez que dos nós estabelecem contato. O valor de

$P_{(a,b)}$ aumenta segundo a Equação (2.1), onde $P_{init} \in [0,1]$ é um valor constante e configurado, que indica uma probabilidade inicial.

$$P_{(a,b)} = P_{(a,b)anterior} + \left(1 - P_{(a,b)anterior}\right) \times P_{init} \tag{2.1}$$

**Diminuição de $P_{(a,b)}$:** Quando dois nós não se encontraram em um intervalo de tempo determinado $t$, que é o número de unidades de tempo que transcorreram; utiliza-se uma constante de envelhecimento $k \in [0,1)$ para diminuir o valor da métrica segundo a Equação (2.2):

$$P_{(a,b)} = P_{(a,b)anterior} \times k^t \tag{2.2}$$

**Propriedade de transitividade:** Esta propriedade estabelece que quando um nó $x$ se encontra frequentemente com um nó $y$, e o nó $y$ com o nó $z$, então é provável que o nó $x$ e o nó $z$ também se encontrem. Deste modo, o nó $x$ pode ser uma boa escolha como nó retransmissor de mensagens para o destino $z$. Na Equação (2.3) mostra-se a representação matemática desta propriedade, onde $\beta \in [0,1]$ é uma constante que define o impacto da regra de transitividade na métrica.

$$P_{(a,c)} = P_{(a,c)anterior} + \left(1 - P_{(a,c)anterior}\right) \times P_{(a,b)} \times P_{(b,c)} \times \beta \tag{2.3}$$

**Retransmissão da cópia:** Quando dois nós $x$ e $y$ se encontram e, o nó $x$ tem uma mensagem $m_z$ com destino ao nó $z$; o nó $x$ só repassa uma cópia de $m_z$ ao nó $y$ se a probabilidade de entrega de $y$ ao nó $z$ é maior do que a probabilidade de entrega do nó $x$ ao nó $z$. Ou seja, o nó $x$ repassa uma cópia de $m_z$ ao nó $y$ se cumpre-se que:

$$P(x,z) < P(y,z). \tag{2.4}$$

Alguns algoritmos de roteamento baseados em utilidade calculam a probabilidade de um evento acontecer para determinar o valor de utilidade de um nó como (LINDGREN; DORIA; SCHELÉN, 2003), (BURGESS et al., 2006). No entanto, estes valores determinam a ocorrência de eventos sem importar o acontecimento ou a relação com outros eventos. Segundo (AHMED; KANHERE, 2010), a utilização de probabilidades condicionais, que determinam a probabilidade de um evento acontecer segundo a ocorrência de eventos relacionados, pode fornecer maior informação para realizar melhores decisões de roteamento. Em razão disto, (AHMED; KANHERE, 2010) usa algoritmos de classificação para a predição

de eventos, utilizando informação de certos acontecimentos. Na seguinte seção, explica-se como os pesquisadores vêm aplicando algoritmos de classificação para a disseminação e roteamento de mensagens.

### 2.2.3 Algoritmos baseados em classificadores

Técnicas de mineração de dados tais como a classificação, podem ser aplicadas para extrair dados de eventos de transmissão e do estado da rede. Estas informações podem ser utilizadas como suporte para os algoritmos de roteamento de mensagens.

A tarefa de classificação consiste em atribuir objetos a uma dentre várias categorias. Formalmente (TAN; STEINBACH; KUMAR, 2005), classificação pode ser definida como a tarefa de aprender uma função alvo $f$ que mapeia cada conjunto de atributos $x$ a um dos atributos de anotação de classe $y$ pré-definidos. Para obter a função alvo $f$ utiliza-se um classificador. O classificador constrói um modelo de classificação a partir de um conjunto de dados de entrada, utilizando um algoritmo de aprendizado. É desejável que o modelo aprendido tenha uma boa capacidade de generalização. Ou seja, um conjunto de atributos $x$, ainda não visto durante o processo de aprendizado, deve ter seu atributo de anotação de classe $y$ atribuído de maneira correta.

No contexto de redes *MANETs* os classificadores vêm sendo utilizados geralmente para a detecção de ataques ou intrusões de segurança (MITROKOTSA; DIMITRAKAKIS, 2013). Não obstante, já alguns anos, realizaram-se alguns estudos para aplicar modelos de classificação para predizer o acontecimento de diferentes eventos na disseminação (COLAGROSSO, 2005) e roteamento (AHMED; KANHERE, 2010) de mensagens.

Em (COLAGROSSO, 2005) propõe-se um algoritmo para diminuir a congestão e o número de colisões do médio de transmissão na disseminação de *broadcasts*. Assim, determinaram-se dois valores para o atributo de anotação de classes para eventos de retransmissão, $y = \{com\ sucesso, sem\ sucesso\}$. Cada tupla coletada em eventos passados define baixo que combinações de conjunto de atributos $x$, um nó realizou uma retransmissão de *broadcast* ***com sucesso*** ou ***sem sucesso***. Na Figura (2.6) são mostradas as tuplas anotadas coletadas no processo de treinamento do classificador do nó $A$. O primeiro caso representa uma retransmissão ***com sucesso***. No segundo caso, dois nós tentam retransmitir ao mesmo tempo, ocasionando um evento de retransmissão ***sem sucesso***. Em futuros eventos, tuplas não anotadas, formadas por um conjunto de atributos $x$ , utilizam o modelo de classificação gerado pelo treinamento do classificador, para predizer se um nó terá sucesso ou não na retransmissão. Com base na predição do classificador o nó decide

retransmitir ou não o *broadcast*. Este procedimento é ilustrado na Figura (2.7).

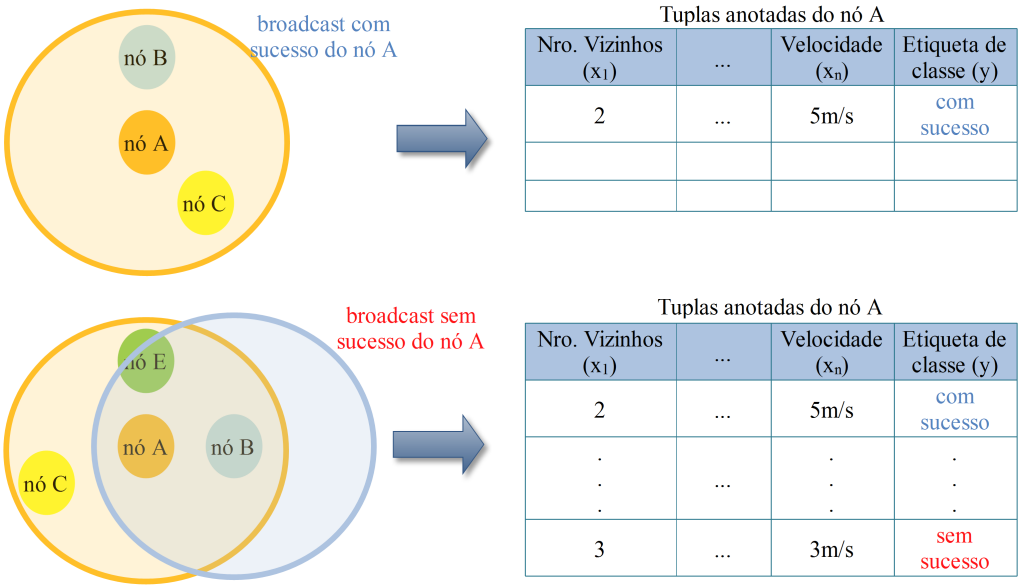



**Figura 2.6: Exemplo de coleta de tuplas anotadas para treinamento do classificador em *MANETs*.**

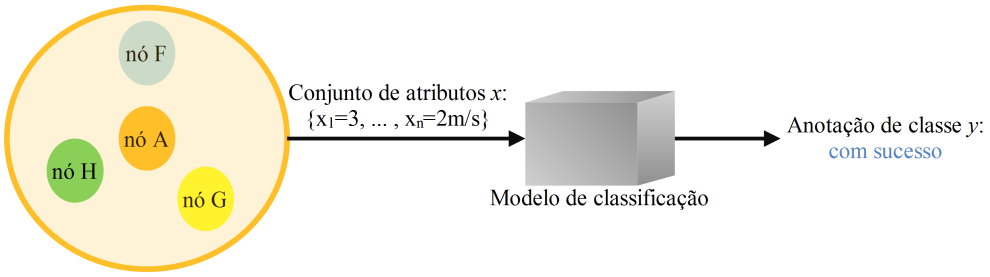



**Figura 2.7: Decisão de retransmissão baseada na predição de um modelo de classificador.**

Existem vários algoritmos de classificação. Em (AHMED; KANHERE, 2010) e (COLAGROSSO, 2005) foi utilizado o classificador *Naive Bayes* que realiza a tarefa de classificação modelando a relação probabilística entre o conjunto de atributos e a variável de anotação de classe. Para tanto, utiliza-se o teorema de *Bayes*, que no contexto de classificação pode ser descrito pela Equação (2.5); onde $X$ denota o conjunto de atributos $x_i$, sendo $i$ o i-ésimo atributo, e $Y$ denota a variável de anotação de classe. Com isso, uma tupla descrita por um conjunto de atributos $X'$, pode ter o valor de sua variável de anotação de classe estimada calculando a probabilidade $P(Y|x'_1, \cdots, x'_n)$ por meio do teorema de *Bayes*

como mostrado na Equação 2.5 e para um número de $n$ atributos.

$$P(Y|x_1,\cdots,x_n) = \frac{P(Y)\cdot P(x_1,\cdots,x_n|Y)}{P(x_1,\cdots,x_n)} \tag{2.5}$$

Para estimar a probabilidade condicional $P(x_1,\cdots,x_n|Y)$ é necessário que no conjunto de treinamento existam tuplas que assumam todas as possíveis combinações de valores de atributos. No entanto, conforme se aumenta os números de atributos, o número de combinações de valores de atributos cresce exponencialmente. Para lidar com este problema, o classificador *Naive Bayes* utiliza a hipótese simplificadora de que os atributos $x_i$ são independentes entre si. Com isso, a equação (2.5) pode ser reescrita como:

$$P(Y|x_1,\cdots,x_n) = \frac{P(Y)\cdot\prod_{i=1}^{n} P(x_i|Y)}{\prod_{i=1}^{n} P(x_i)} \tag{2.6}$$

Isso ocorre porque, para variáveis aleatórias independentes $x_1,\cdots,x_n$, têm-se que:

$$\begin{aligned} P(x_1,\cdots,x_n) &= P(x_1)\cdots P(x_i) \\ P(x_1,\cdots,x_n|Y) &= P(x_1|Y)\cdots P(x_i|Y) \end{aligned} \tag{2.7}$$

Notando que o denominador na equação 2.6 é invariante quanto a mudança da variável de classe $Y$, pode-se ignorá-lo, resultando na expressão utilizada pelo classificador *Naive Bayes* para estimar a probabilidade condicional de classe na Equação (2.8):

$$P(Y|x_1,\cdots,x_n) = P(Y)\cdot\prod_{i=1}^{n} P(x_i|Y) \tag{2.8}$$

Neste projeto de mestrado foram utilizados classificadores baseados em árvores de decisão para realizar o roteamento de mensagens em *VANETs*. Especificamente, foi utilizado o algoritmo *C 4.5* para realizar o treinamento das árvores de decisão (QUINLAN, 1993). Esta escolha está justificada no bom desempenho do *C 4.5* tanto em precisão e porcentagem de acertos, como também em tempo de treinamento de predição (LIM; LOH; SHIH, 2000). Adicionalmente, o algoritmo *C 4.5* quando comparado ao *Naive Bayes*, apresentou maior precisão no processo de classificação para aplicações em *VANETs* deste projeto. A continuação apresenta-se alguns conceitos básicos referentes a este classificador.

#### 2.2.3.1 Algoritmos baseado em árvore de decisão C4.5

Os classificador *C 4.5* utiliza uma estrutura de árvore de decisão para realizar a classificação de uma tupla não anotada. Uma árvore de decisão é uma estrutura de dados na qual os nós internos representam testes sobre atributos e os nós folhas representam atributos de anotação de classe.

Vamos supor um caso hipotético no qual queremos criar uma árvore para decidir se uma mensagem deve ser enviada ou não. Neste caso teremos dois possíveis valores para o atributo de anotação de classe: ***enviar*** e ***não enviar***. Considera-se a velocidade do nó $(m/s)$ e número de vizinhos como atributos. Na Figura (2.8) é mostrado um exemplo de árvore de decisão para a classificação de tuplas não anotadas. O processo de classificação consiste em percorrer a árvore desde o nó raiz até um nó folha. Vamos supor que se precisa decidir se uma mensagem deve ser enviada ou não e os atributos são os seguintes: velocidade $= 6m/s$, número de vizinhos $= 3$. Neste caso, o teste realizado no nó raiz (velocidade $\geq 5$) nós levaria ao nó filho da direita. O nó da direita é um nó folha que representa o ou valor de atributo de anotação de classe ***enviar***. Desta maneira o resultado da classe retornada é ***enviar***. Vamos percorrer agora a árvore de decisão da Figura (2.8) para um nó com os seguintes atributos: velocidade $= 2m/s$, número de vizinhos $= 4$. Neste caso, o teste realizado no nó raiz nós levaria ao nó filho da esquerda, que representa uma decisão sobre o atributo número de vizinhos. Uma vez que o número de vizinhos $= 4 \geq 3$ chegamos ao nó folha com a classe ***não enviar***.

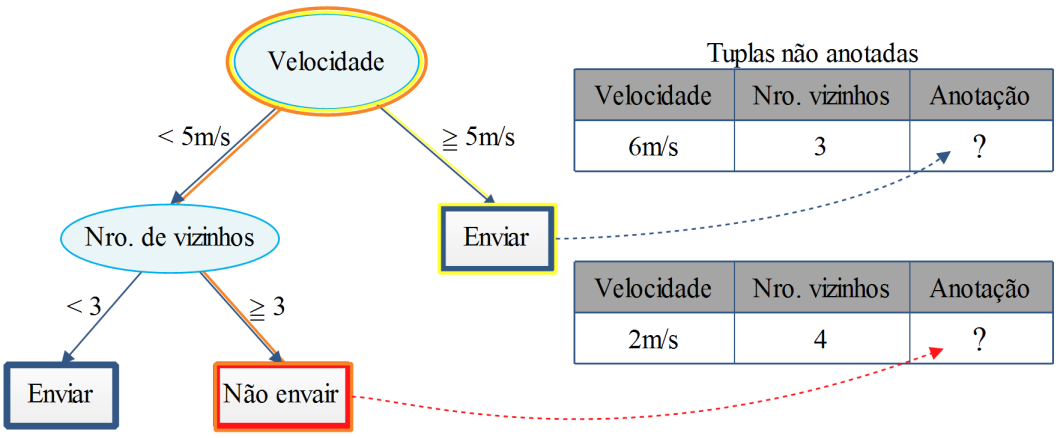



**Figura 2.8: Exemplo de árvore de decisão e classificação de tuplas não anotadas.**

O treinamento do classificador *C*4.5 consiste em gerar uma árvore de decisão. Para tanto, inicialmente todas as tuplas anotadas do conjunto de treinamento são atribuídas a um único nó raiz. Este nó raiz é dividido recursivamente por meio de decisões baseadas

nos atributos até que um critério de parada seja atingido. Desta maneira, o algoritmo de treinamento consiste então em duas etapas principais:

- Etapa1: Analisar o nó. Se o critério de parada foi atingido, parar a divisão.
- Etapa2: Se o critério de parada não foi atingido, realizar uma divisão do nó por meio de um teste de atributo. Aplicar o algoritmo recursivamente aos nós filhos gerados pela decisão.

Desta maneira, o algoritmo de treinamento do $C4.5$ apresenta dois pontos principais: determinação do critério de parada e como encontrar uma condição que irá realizar a divisão de um nó. O critério de parada consiste em verificar se todas as tuplas do conjunto de treinamento atribuídas ao nó pertencem à mesma classe $C_i$ ou se o número de tuplas é pequeno (valor estabelecido na configuração do classificador). Se essa condição for verdadeira, então o nó é um nó folha e sua classe é $C_i$.

Para encontrar a condição que irá realizar a divisão de um nó da árvore de decisão o $C4.5$ encontra dentre todas as divisões candidatas, aquela que maximize a pureza dos nós filhos. Um nó puro é aquele no qual quase todas as tuplas pertencem à mesma classe. Existem várias medidas para quantificar a pureza de um nó. Uma delas é a medida de entropia da informação. Sendo $p(i)$ a fração de tuplas que pertencem a classe $i$ e $C = \{C_1, C_2, \cdots, C_N\}$ o conjunto de todas as classes, temos que a medida de impureza $I$ de um nó é dada por:

$$I = 1 - \sum_{i \in C} [p(i)^2] = 1 - p(C_1)^2 + p(C_2)^2 + \cdots + p(C_N)^2 \tag{2.9}$$

## 2.3 Caracterização da topologia de rede para o projeto de algoritmos de roteamento

O desempenho dos algoritmos de roteamento está sujeito à conectividade entre os nós, que depende do seu movimento. Em razão disto, a representação das propriedades de conectividade de uma rede, cujos nós têm um movimento particular, determina o impacto que o movimento dos nós tem na estrutura da rede, e consequentemente, no desempenho dos algoritmos de roteamento.

As métricas utilizadas, para a caracterização das propriedades de conectividade de uma rede, são expressões matemáticas relacionadas principalmente ao estado do enlace

entre pares de nós e, métricas próprias da teoria de grafos para abstrair o estado da topologia de rede e o seu comportamento no transcurso do tempo.

A terminologia usada para as métricas é a seguinte:

- $G$, representa o grafo de conectividade entre os nós da rede.
- $V = \{v_i\}$, representa o conjunto dos nós, onde $i = \{1,2,...,N\}$, e $N$ é igual ao número de nós na rede.
- $E = \{e_{ij}\}$, representa o conjunto de enlaces de comunicação entre os nós $v_i$ e $v_j$, que pertencem ao conjunto $V$; sendo $i \neq j$.
- $G(t)$, $V(t)$ e $E(t)$, representam respectivamente o estado do grafo de conectividade $G$, do conjunto de nós $V$; e do conjunto de enlaces $E$, no tempo $t$.
- $X(i,j,t)$: Indicador de conectividade, onde $X(i,j,t) = 1$ se existe um enlace entre os nós $v_i$ e $v_j$ no tempo $t$ e 0 se em caso contrário. Além, $X(i,j) = max_{t=1}^{T} X(i,j,t)$, sendo $T$ o tempo de simulação.
- $k_i$, é o grau de conectividade de o nó $v_i$; ou seja, é o número de conexões que possui.
- $n(k)$, é o número de nós com grau $k$.

As métricas incluídas nesta seção estão divididas em dois grupos: (i) métricas para o estudo da conectividade entre pares e, (ii) métricas de teoria de grafos para o estudo da conectividade de rede.

### 2.3.1 Métricas para o estudo da conectividade entre pares

Dentro deste grupo incluem-se quatro métricas: (i) número de mudanças do estado do enlace, (ii) duração do enlace, (iii) tempo de contato e, (iv) tempo entre contatos. As duas primeiras métricas foram utilizadas em (BAI; SADAGOPAN; HELMY, 2003) para o estudo da estabilidade dos enlaces entre pares de nós. Estas métricas explicam-se a continuação.

- Número de mudanças do estado do enlace ($CE(i,j)$): É o número de transições do estado $DOWN$ para o estado $UP$ do enlace de conectividade entre os dois nós específicos $i$ e $j$. Esta métrica também determina o número de encontros entre os nós $i$ e $j$.

- Duração do enlace ($DE(i,j)$): É a relação do tempo de duração de um enlace entre dois nós $i$ e $j$ considerando o número de mudanças do estado de enlace $CE(i,j)$.
- Tempo de contato: É o tempo que dura um enlace entre pares de nós da rede.
- Tempo entre contatos: É o tempo que transcorre entre a perda de conectividade entre dos nós e o momento em que voltam a estabelecer conectividade.

Na Tabela (2.3) é mostrada a representação matemática das duas primeiras métricas segundo (BAI; SADAGOPAN; HELMY, 2003).

**Tabela 2.3: Métricas de estabilidade do enlace.**

| | |
|---|---|
| Número de mudanças do estado de enlace | $CE(i,j) = \sum_{t=1}^{T} C(i,j,t)$, onde $C(i,j,t) = 1$ se $X(i,j,t-1) = 0$ e $X(i,j,t) = 1$ |
| Duração do enlace | $DE(i,j) \begin{cases} \frac{\sum_{t=1}^{T} X(i,j,t)}{CE(i,j)} & se\ CE(i,j) \neq 0 \\ \sum_{t=1}^{T} X(i,j,t) & em\ qualquer\ outro\ caso. \end{cases}$ |

## 2.3.2 Abstração da topologia da rede usando teoria de grafos

A teoria de grafos vem sendo utilizada para a caracterização e comparação dos grafos de conectividades das topologias de rede. O conhecimento das propriedades de conectividade, permite o projeto de protocolos mais efetivos para ambientes específicos e a detecção de nós de maior qualidade na topologia da rede (VENDRAMIN et al., 2012). Nesta seção, incluem-se métricas de teoria de grafos utilizadas para abstrair as características da topologia de rede. Dependendo da informação que se necessita para o cálculo das métricas, dividem-se em métricas locais e globais. As métricas locais são aquelas que podem se calcular com informação dos nós e os seus nós vizinhos. As métricas globais, são aquelas que precisam informação da topologia de rede para o seu cálculo.

**Métricas locais:** O **grau de conectividade** $k$ é a propriedade mais básica da teoria de grafos. Existem métricas baseadas no estudo do grau de conectividade que podem caracterizar diferentes topologias de modelos de redes comuns, como descrito em (WEHRLE; GROSS; GÜNES, 2010). A métrica comumente utilizada para caracterização de modelos de rede é a **distribuição do grau de conectividade** $P(k)$, que calcula-se como :

$$P(k) = \frac{n(k)}{|V|} \tag{2.10}$$

A **média de do grau de conectividade** $\overline{k}$; pode se derivar da distribuição de $P(k)$, ou calcular-se por médio do número de enlaces entre os nós conectados, como representa-se na Equação (2.11)

$$\overline{k} = \frac{2|E|}{|V|} = \sum_{k=0}^{k_{max}} k \cdot P(k) \tag{2.11}$$

Algumas redes tendem de apresentar alguma correlação entre o grau dos nós. Neste tipo de redes, como as redes sócias, uma métrica comumente utilizada é a métrica de ***joint degree distribution*** ou ***JDD*** que representa a probabilidade de existir um enlace que conecte dois nós de grau $k$ e $k'$, ou seja, $P(k,k')$. Uma forma de expressar o *JDD* é relacioná-lo com a probabilidade condicional de que dado que exista um nó com grau $k'$, exista um nó conectado a ele com grau $k$; isto é $P(k \mid k')$, tendo em conta que para grafos não dirigidos $P(k \mid k')$ é igual a $P(k' \mid k)$ . O valor de $P(k \mid k')$ pode calcular-se segundo a Equação (2.12) (COSTA et al., 2006).

$$P(k \mid k') = \frac{\overline{k} \cdot P(k',k)}{k' \cdot P(k')} \tag{2.12}$$

O ***lobby índex***, $L_{inx}$, é uma métrica proposta por (KORN; SCHUBERT; TELCS, 2009) para aproximar o valor de centralidade dos nós, que reflexa a importância de cada nó dentro da rede. Posteriormente em (PALLIS et al., 2009) afirma-se que está métrica sim pode identificar nós de boa qualidade no contexto de conectividade. A vantagem de aproximar o valor de centralidade com o valor do *lobby índex* é que um nó só precisa informação do número de vizinhos dos seus vizinhos, diferentemente das métricas de centralidade que precisam conhecer a topologia global da rede. Está métrica está inspirada no conhecido *h-index*, assim, o $L_{inx}$ de um nó $v_i$ é o maior valor inteiro $l$ tal que o nó $v_i$ tem pelo menos $l$ vizinhos (nós conectados no primeiro salto) com grau de pelo menos $l$. Por exemplo, na imagem (2.9) o $L_{inx}$ do nó 8 é 2 porque tem mais de dois vizinhos com grau 2.

**Métricas globais da rede:** Estão baseadas principalmente no valor do caminho mais curto entre nós (*shortest paths*), também conhecido como distância. A distância entre dois nós $v_i$ e $v_j$, ou $sp(v_i,v_j)$; é o caminho com menor longitude entre eles. A longitude ($L_{sp}$) é a suma dos valores ou pesos dos enlaces que interconectam os pares de nós que formam o caminho. Para grafos com enlaces sem pesos, o valor para cada enlace pode se estabelecer em uma unidade, desta maneira, a longitude da distância seria o número de enlaces que conformam o caminho. No contexto

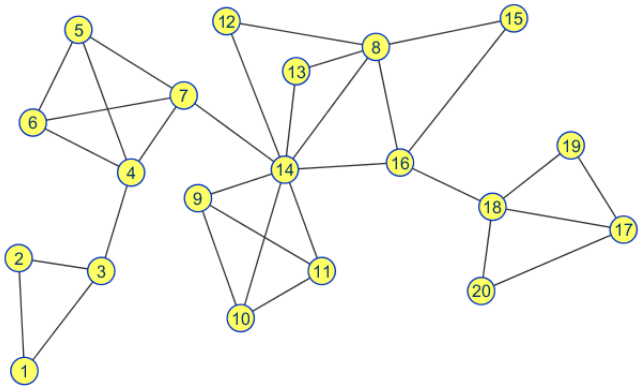



**Figura 2.9: Exemplo de *lobby índex*.**

de redes, a distância é o número de saltos entre dois nós. O diâmetro da rede é a distância de maior longitude que conecte a qualquer par de nós da rede. Se a rede for desconexa, o diâmetro e o maior diâmetro dos componentes conectados da rede.

O estudo das distâncias ou caminhos mais curtos, pode revelar propriedades de *small world*, como em redes sociais. No caso das *VANETs*, que não apresentam estas propriedades, esta informação pode se usar em métricas de centralidade para determinar que nós são importantes dentro dos componentes de rede. As métricas mais utilizadas para o estudo de centralidade segundo (BRANDES, 2001) são:

- *Stress centrality* ($C_S$), é o número distâncias entre todos os pares de nós da rede ($\sigma$) que passam por um nó. O valor de $C_S$ de um nó $v_i$, é calculado segundo a Equação (2.13).

$$C_S(v_i) = \sum_{v_s \neq v_i \neq v_t} \sigma(v_s, v_t), \; onde \; v_s, v_i, v_t \in V \tag{2.13}$$

- *Betweenness centrality* ($C_B$), é a relação de o número de distâncias entre os pares de nós da rede ($\sigma$) que passam por um nó, do total de distâncias existentes na rede. Esta métrica favorece mais a nós que unem comunidades de nós do que a nós dentro de comunidades. Uma rede apresenta estrutura de comunidades se estiver formada por grupos de nós altamente conectados entre eles. Um grupo de nós é considerado uma comunidade se o número de enlaces formados dentro dele for maior ao número de enlaces que o interconecta com outras comunidades. O valor de $C_B$ de um nó $v_i$, é calculado segundo a Equação (2.14).

$$C_B(v_i) = \sum_{v_s \neq v_i \neq v_t} \frac{\sigma_{v_s,v_t}(v_i)}{\sigma_{v_s,v_t}}, \; onde \; v_s, v_i, v_t \in V \tag{2.14}$$

- *Closeness centrality* ($C_C$)(NEWMAN, 2005), é a inversa da soma das distancias de um nó com todos os outros nós da rede. É calculado segundo a Equação (2.15). Está métrica apresenta certos inconvenientes quando se aplica em redes desconexas em razão de que considera as conexões de um nó com todos os outros nós da rede. Quando não existir conexões entre pares, a distância é considerado $\infty$. Esta limitação faz com que pesquisadores meçam a métrica dentro dos componentes. Porém, nós com maior *closeness* em componentes pequenos podem ser favorecidos com esta métrica.

$$C_C(v_i) = \frac{1}{\sum_{j \in V} L_{sp}(v_i, v_j)} \tag{2.15}$$

Os algoritmos de roteamento vêm incorporando métricas de centralidade para as decisões de encaminhamento de mensagens em *DTNs*, a desvantagem de estes algoritmos, é que precisa-se da derivação dos grafos de conectividade, que é um processo muito complexo.

# Capítulo 3

## Simulação e caracterização dos traces de movimento

*Este capítulo foi incluído com os objetivos de descrever o processo de simulação, assim como diferenciar e reconhecer a importância da utilização dos* traces *de movimento real contra os modelos sintéticos. Considera-se útil a informação específica do simulador* The ONE*, uma vez que este simulador está sendo muito utilizado nos últimos anos.*

## 3.1 Introdução

A simulação de cenários para a avaliação do desempenho dos algoritmos de roteamento em redes móveis está formada por dois componentes (HARTENSTEIN; LABERTEAUX; EBRARY, 2010): (i) a simulação do movimento dos nós e, (ii) a simulação do comportamento de *networking*. A relação e a interação entre ambas as partes depende do tipo de aplicação que se deseja reproduzir. Por exemplo, se a aplicação está dirigida ao estudo de aplicações de eficiência e auxilio do tráfego veicular, deve existir uma retroalimentação entre ambos os componentes a fim de que seja possível modificar a escolha de rotas de tráfego, segundo a informação obtida da aplicação. No caso da avaliação de algoritmos de roteamento, um simulador de rede que forneça um simulador de movimento incluído é ideal porque poderia abastecer de informação do movimento para aprimorar o desempenho dos protocolos.

Em razão de que o desempenho dos algoritmos de roteamento em redes móveis está altamente influenciado pelos modelos ou padrões de movimento dos nós, é muito importante usar uma representação do movimento ou mais próxima à realidade. Existem diferentes

modelos de movimento sintéticos que poderiam se usar para representar e reproduzir o movimento dos veículos nos simuladores. Alguns exemplos são: movimentos aleatórios, do caminho mais curto, do caminho no menor tempo (segundo a velocidade das possíveis rotas), baseado em rotas estabelecidas (*paths*), etc. A vantagem dos modelos sintéticos é que permitem criar variedade de casos de estudo, manipulando as localizações dos nós e controlando facilmente a densidade de nós. O problema dos modelos sintéticos é que resumem as principais características dos movimentos sem conseguir representar totalmente o movimento real. Por esta razão, diferentes organizações fornecem conjuntos de dados com localizações geográficas de veículos, e outros tipos de nós, movimentando-se em ambientes reais (urbano ou rural). Algumas organizações também disponibilizam *testbeds* de veículos reais para acessar remotamente e realizar a coleta de dados. Esta informação de movimento é conhecida como *traces* de movimento e podem se conseguir através de fontes como: Cabspot, MobiLib, Crawdad, Dieselnet e MIT Reality Mining.

## 3.2 O simulador *The ONE*

*The Opportunistic Networking Environment (ONE) Simulator* (KERäNEN; OTT; KäRKKäINEN, 2009), (KERäNEN; KäRKKäINEN; OTT, 2010), é um simulador para a avaliação de algoritmos de roteamento e de aplicação em *DTNs*. Foi criado no Departamento de Comunicações e *Networking* da Universidade de Tecnologia de *Helsinki*. Este simulador vem sendo utilizado nos últimos anos em pesquisas sobre *DTNs* como: (EKMAN et al., 2008), (MOGHADAM; SCHULZRINNE, 2009), (YIN; LU; CAO, 2009), (GAO; ZHANG; ZHANG, 2010), (YU; WU; HU, 2010), (GUNASEKARAN; MAHENDRAN; MURTHY, 2012). O simulador está desenvolvido em *Java* e realiza tanto simulação de movimento como de *networking*. O simulador tem a capacidade de reproduzir *traces* externos, que são arquivos que contém tuplas de dados da ocorrência de eventos em determinados tempos. Um exemplo de *traces* externo é o *trace* de movimento. Os *traces* que podem se reproduzir no simulador são de eventos de mensagens, de conectividade e de movimento.

Os componentes principais e o funcionamento geral do simulador resumem-se na Figura (3.1). Os componentes do simulador, explicam-se a continuação:

**Motor de simulação:** Realiza a execução de diferentes tarefas do processo de simulação.

**Repórter:** Realiza o cadastro dos resultados finais das métricas de desempenho das simulações. Os relatórios fornecidos também podem registrar a ocorrência de eventos em tempo de simulação (eventos de movimento, de mensagens, roteamento ou de

conectividade). A programação de novos relatórios, baseados nas *interfaces* e relatórios do simulador, permite formatar o conteúdo para outros programas, estadísticos ou de visualização, como *GraphViz*.

**Módulo de movimentos:** Reúne um conjunto de modelos de movimento, que podem ser sintéticos ou modelos para a reprodução de *traces* de movimento real. A execução dos modelos de movimento pelo motor de simulação fornece dados de conectividade, que são resultado de eventos de conectividade que ocorreram pela mobilidade dos nós dentro do raio de alcance de outros nós. Opcionalmente estes eventos podem ser cadastrados em relatórios para o posterior análises de dados o para a criação de *traces* de conectividade.

**Gerador de eventos:** É utilizado para agendar a ocorrência de alguns eventos. Geralmente, é usado para gerar eventos de criação de mensagens com uma origem e destino determinados. Também pode ser utilizado para criar eventos em nível de camada de aplicação. O motor de execução pode utilizar o gerador interno (*MessageEventGenerator*, que cria novos mensagens em intervalos de tempo determinados, ou utilizar os geradores externos que criam eventos segundo as especificações de arquivos de texto.

**Módulo de roteamento:** Contém algoritmos de roteamento muito referenciados para *DTNs*: *Epidemic*, *Prophet*, *Spray And Wait*, *Maxprop*, entre outros. Adicionalmente inclui o modelo de roteamento *PassiveRouter* que anula as tarefas de roteamento do simulador. Deste modo o simulador age só como simulador de movimento e exporta os eventos de movimento e conectividade para outros simuladores de roteamento, como: *NS-2*[1] e *dtnsims2*. Atualmente o simulador fornece um*script* para trazer de volta os resultados de simulação do simulador *dtnsims2* para a geração de relatórios e assim a finalização do ciclo de simulação.

### 3.2.1 Funcionamento do simulador

O simulador utiliza um arquivo de configuração, cujo *script* determina várias propriedades: do ambiente, do cenário de simulação, da mobilidade, dos nós, de rede, das localizações dos arquivos de dados que se precisam, entre outros. Os arquivos de dados que se utilizam podem corresponder a: mapas ou rotas em formato *wkt* (*Well-known*

[1] http://www.nsnam.org/

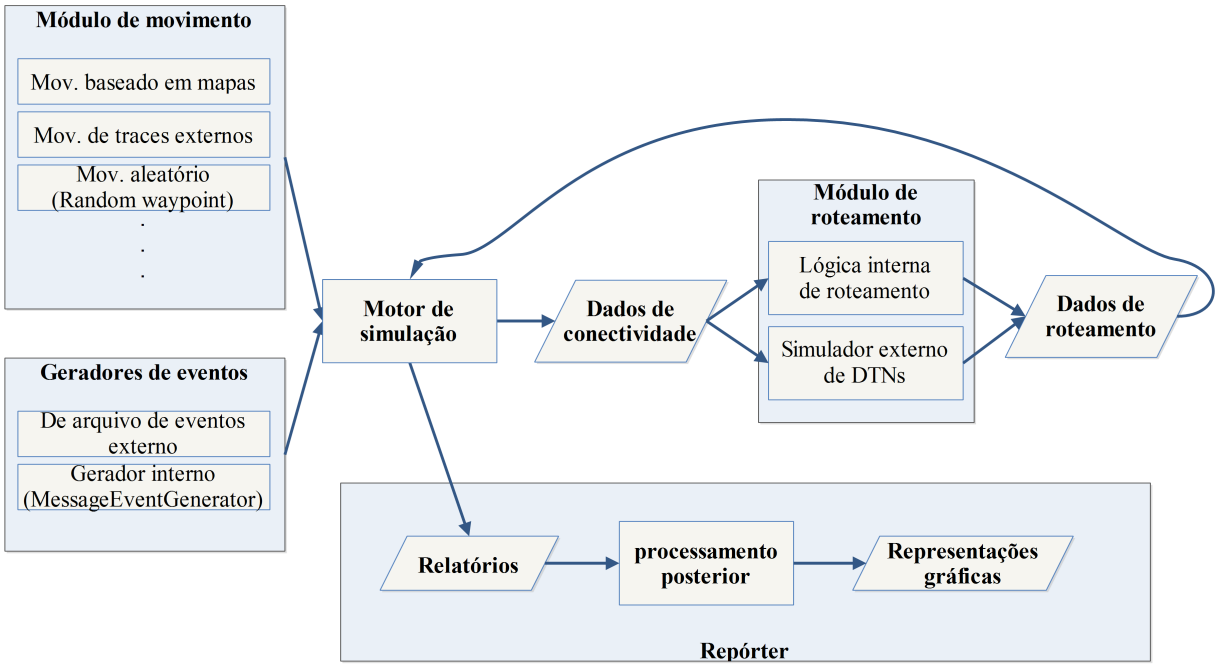



**Figura 3.1: Entorno de simulação**

*text*), *traces* de movimento, arquivos de eventos externos (eventos de conectividade, eventos de mensagem), imagens de mapas (opcionalmente), entre outros segundo o tipo de movimento.

As especificações dos parâmetros de simulação são indicadas em arquivos de configuração. Estes arquivos incluem tuplas da forma: $namespace.key = v$, onde o *namespace* indica o âmbito ou nome da classe para a qual está se configurando a propriedade *key* com o valor $v$. Um parâmetro importante na configuração é o valor de intervalo de atualização (*Scenario.updateInterval*). O simulador realiza a simulação progressivamente em cada intervalo de atualização, isto é, em cada intervalo de atualização os objetos da simulação são atualizados como mínimo uma vez. A menor intervalo de atualização, maior precisão na simulação, maior consumo de memória *RAM* e tempo de execução.

O arquivo de configuração por padrão é o arquivo *default_settings.txt*, este arquivo é sempre considerado na leitura dos parâmetros de simulação. Esta função é útil para especificar os parâmetros básicos e constantes no arquivo de configuração padrão e adicionar parâmetros variáveis que serão incorporados adicionalmente em arquivos de configuração adicionais.

**Execução de vários cenários por indexação:** Usualmente, os algoritmos de roteamento utilizam parâmetros especiais que determinam o seu comportamento. É por

isto, que se precisa testar com vários valores de parâmetros para revelar o comportamento do algoritmo. Para este propósito é melhor a execução do simulador em modo de consola e não em modo *GUI*. Em modo consola o simulador permite configurar propriedades com vários valores, que representam diferentes cenários de simulação; e executar várias simulações em lote utilizando os valores indexados por o número de simulação atual. Este modo de execução é conhecido como *run indexing*, ou execução por indexação, e permite especificar várias simulações em um só arquivo de configuração. Na Figura (3.2) são mostrados dois exemplos de como executa-se o simulador usando *run indexing*. No primeiro exemplo, em cada simulação de um experimento, o algoritmo de roteamento cambiará de valor de propriedade dependendo do valor que indexa o número de simulação. Na primeira execução o simulador usará o valor de *EpidemicRouter* para a propriedade *Group.router*, na segunda simulação o valor de *ProphetRouter* e finalizará com uma terceira simulação usando o valor de *SprayAndWait*. Quando utilizar-se mais de um valor para mais de uma propriedade, o simulador realiza todas as possíveis simulações segundo as possíveis combinações de valores como se mostra no segundo exemplo da Figura (3.2). No segundo caso, se executarão seis simulações com os valores das possíveis combinações: (EpidemicRouter,50), (ProphetRouter,70) (SprayAndWaitRouter,50) (EpidemicRouter,70) (ProphetRouter,50) (SprayAndWaitRouter,50).

Run indexig para um parâmetro:

No arquivo de configuração arquivo.txt:

*Group.router = [EpidemicRouter; ProphetRouter; SprayAndWaitRouter]*

No comando de execução:

*./one.sh -b 3 arquivo.txt*

Run indexig para mais de um parâmetro:

No arquivo de configuração arquivo.txt:

*Group.router = [EpidemicRouter; ProphetRouter; SprayAndWaitRouter]*
*Group.nrofHosts = [50; 70]*

No comando de execução:

*./one.sh -b 6 arquivo.txt*

**Figura 3.2: Execução por indexação de vários cenários.**

Desde o ponto de vista do desenvolvedor é necessário conhecer o funcionamento interno e as classes más importantes envolvidas no processo de simulação. Está informação é útil para o desenvolvimento de novos algoritmos de roteamento, novos relatórios, novos parâmetros de simulação ou novas funcionalidades em nível de aplicação. No Anexo (A)

explica-se o processo de simulação dividido em três fases para melhor explicação utilizando alguns diagramas de sequencia que incluem só as classes e operações consideradas de maior relevância para o usuário desenvolvedor.

## 3.3 Recursos utilizados e modificações no simulador

### 3.3.1 Modelo de movimento externo

O modelo de movimento utilizado do simulador é o modelo *ExternalPathMovement*. Este modelo de movimento usa *traces* de movimento externos em formato de rotas formadas por pontos de localizações e tempos. Utiliza dois arquivos de dados: (i) o **arquivo de rotas**, que inclui tuplas de: *nó* $tempo_1, x_1, y_1$ $tempo_2, x_2, y_2 \cdots tempo_n, x_n, y_n$ e, (ii) o **arquivo de atividade de movimento**, que inclui tuplas de intervalos de tempo em que os nós se movimentaram: *nó* $tempo_{inicio}$ $tempo_{fim}$.

**Traces de movimento:** Os *traces* de movimento real utilizados foram fornecidos por CRAWDAD [2]. O conjunto de dados contém informação das localizações, em intervalos de tempo pequenos, de mais de 1000 ônibus do condado de *King* em *Seattle*, *Washington*[3]. Estes *traces* foram utilizados previamente em (JETCHEVA et al., 2003) para o estudo de *MANETs* e em (AHMED; KANHERE, 2010) para o estudo de *DTNs*. Na Figura (3.3)) mostra-se o mapa geográfico e de rotas, que envolve uma área aproximadamente de 55Km. x 83Km. e mais de 200 rotas de transporte.

O material utilizado para a representação do movimento dos veículos, corresponde a um subconjunto dos *traces* de movimento. Considera-se um subconjunto dos dados porque o tempo de execução das simulações incrementar-se-ia consideravelmente de considerar o conjunto total. A amostra utilizada é de uma área aproximada de 144 $Km^2$, que inclui a área de maior interesse segundo a densidade de nós. Os movimentos incluídos correspondem à faixa de tempo das 7 horas da manhã até as 12 horas (cinco horas de movimento). Na Seção (3.4) determina-se as características de conectividade encontradas, segundo as métricas da Seção (2.3).

**Pre-processamento dos *Traces* de movimento:** Para utilizar os *traces* de movimento com o modelo de movimento *ExternalPathMovement* do simulador, foi necessário

[2] http://crawdad.cs.dartmouth.edu/
[3] http://metro.kingcounty.gov/oltools/tracker.html

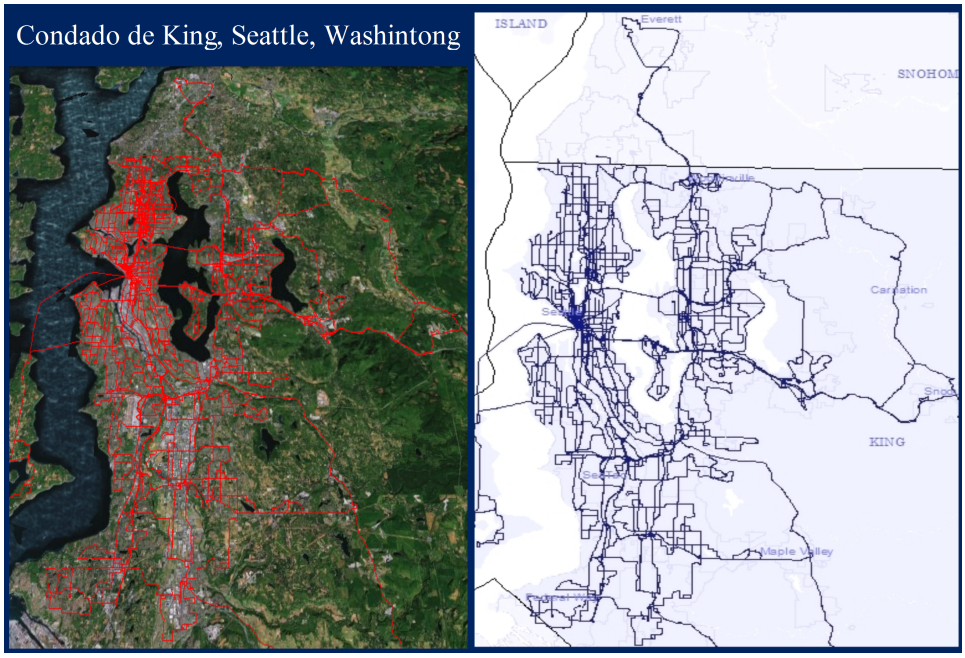



**Figura 3.3: Mapa geográfico e de rotas correspondentes aos movimentos dos *traces* utilizados.**

realizar um programa de *software* que realize várias tarefas, entre elas, a formatação dos dados aos formatos dos arquivos requeridos pelo simulador. O programa é capaz de formatar todos os movimentos encontrados dentro de uma área e em um intervalo de tempo especificados. Adicionalmente, este programa gera os dois arquivos de dados que utiliza o modelo de movimento (o arquivo de rotas e arquivo de atividade de movimento) e um arquivo com um *script* de configurações básicas para o simulador.

Uma dificuldade apresentada foi como excluir os movimentos dos nós quando saírem da área de estudo e como incluir os movimentos dos nós quando eles voltarem. Este problema se deve a que o simulador representa o movimento como deslocamentos em linha reta de um ponto (ou localização inicial) a outro (localização destino) e, a velocidade de deslocamento, é função da diferencia dos tempos em que o nó apareceu em cada ponto. Na Figura (3.4) é mostrado um exemplo em que um nó esteve por última vez dentro da área de estudo na localização $A$, depois de uma serie de movimentos excluídos fora da área de estudo, o nó volta à área estudada na localização $B$. O simulador representa este caso como um movimento em linha reta desde o pondo onde o nó saiu, até o pondo onde ele voltou. Estes movimentos errados alteram consideravelmente os resultados de desempenho. Encontraram-se duas possíveis soluções: (i) incluir só os nós cujos movimentos realizam-se dentro da área durante as cinco horas de simulação e, (ii) permitir que o simulador represente

os movimentos errados, porém, programando a funcionalidade para que exclua a atividade de *networking* para não influenciar nos resultados. A abordagem (i) tem a vantagem de ser mais simples e foi utilizada com os mesmos *traces* de movimento em (AHMED; KANHERE, 2010), incluindo só 35 nós em uma área limitada. No entanto, uma vez que neste trabalho estudam-se algoritmos de roteamento de múltiplas cópias, para o estudo da sobrecarga de rede é necessário conservar uma densidade de nós aproximada à realidade. É por isto que se optou pela opção (ii), conseguindo uma densidade média de 254 nós e assim cenários mais reais. Assim, uma última tarefa do programa desenvolvido, é de gerar um terceiro arquivo de intervalos de tempos de atividade de rádio para cada nó.

Para não influenciar na avaliação da probabilidade de entrega, só os nós que permaneçam o tempo todo dentro da área (13 nós) poderão criar novas mensagens e ser nós de destino. Considera-se necessária esta restrição porque se um nó destino não estivesse na área em alguns intervalos de tempo, a mensagem não poderá ser entregue e o desempenho dos protocolos avaliados será muito baixo. Apesar desta limitação, os nós que entram e saem da área sim poderão ser nós retransmissores, já que em casos reais, as cópias podem ser transmitidas a veículos que se deslocam fora da área. Para a exclusão da atividade de *networking* em intervalos de tempo, foi necessário realizar algumas mudanças no simulador, as quais descrevem-se no Anexo (A).

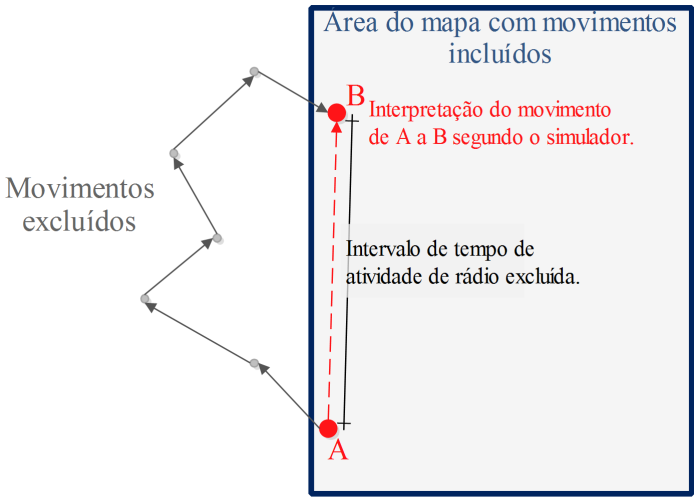



**Figura 3.4: Problema da amostragem dos *traces* de movimento.**

## 3.4 Comparação dos *traces* de movimento com modelos de movimento sintético

Para a captura das características de cada modelo de movimento, realizou-se um conjunto de experimentos e obteve-se as métricas de conectividade da Seção (2.3) através de vários relatórios programados no simulador. Entre vários resultados relacionados à teoria de grafos, inclui os grafos de conectividade em formato de entrada para a ferramenta *CytoScape*[4]. Esta ferramenta foi utilizada para capturar algumas métricas relacionada à teoria de grafos.

### 3.4.1 Cenários de estudo

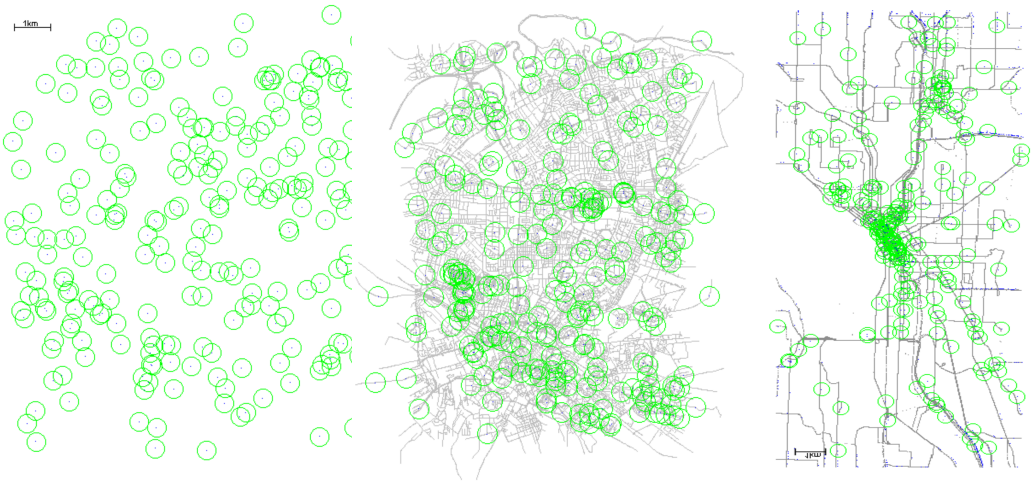
**Modelos de movimentos usados** Os *traces* de movimento foram descritos anteriormente na Seção (3.3.1). A continuação apresentam-se as características dos modelos de movimento sintéticos com os quais foi comparado para a sua diferenciação.

- *Random Waypoint* (*RWP*): É um modelo de movimento sintético onde tanto a localizações dos nós e as velocidades são determinadas aleatoriamente. Cada nó movimenta-se independentemente dos outros ao longo de uma linha a partir de um ponto aleatório $p_i$ da área de movimentação, para o próximo ponto $p_{i+1}$. Este comportamento pode gerar movimento em zigue-zague de linhas retas entre pontos. O cenário configurado para este modelo de movimento é de uma área de movimentação de 139 $Km^2$.
- *Shortest path map based movement* (*SPMB*): é uma combinação entre o *RWP*, no sentido que a escolha dos pontos de destino é realizada aleatoriamente, e os movimentos baseados em mapa que usam *Dijkstra* para encontrar o caminho mais curto no deslocamento de um ponto a outro. Este movimento apresenta maiores restrições de mobilidade que o *RWP* porque o movimentos dos nós está limitados a seguir determinadas rotas. O mapa utilizado é da cidade de *Manchester*, mapa capturado com a ferramenta *Open Street Map*. Para utilizar este mapa, foi convertido ao formato *wkt*, que é o formato que utiliza o simulador para a representação de rotas e mapas. A área total de movimentação configurada para este cenário é aproximadamente de $127Km^2$.

Na Figura (3.5) mostra-se a distribuição dos nós na área de movimentação nos diferentes modelos de movimento. No caso dos *traces* de movimento, aprecia-se

[4]http://www.cytoscape.org/

ademais as zonas de maior concentração de nós.

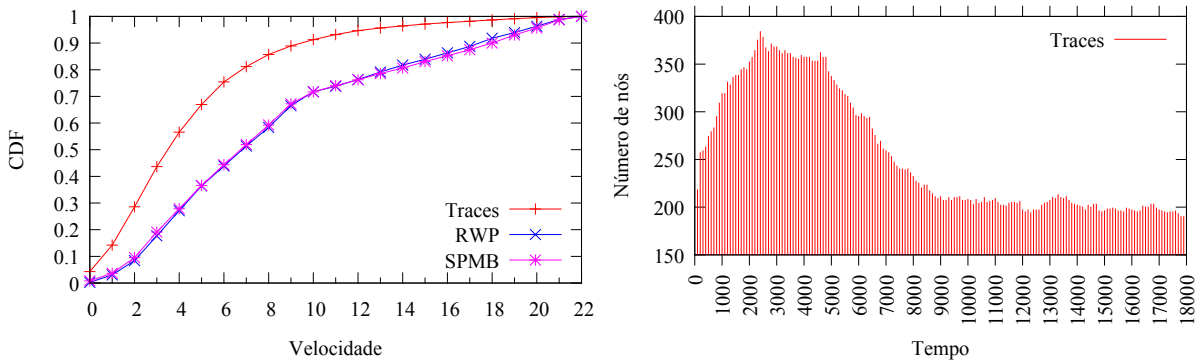


(a) *RWP* (b) *SPMB* (c) *Traces*

**Figura 3.5: Distribuição dos nós na área de movimento nos cenários com diferentes modelos de movimento.**

**Parâmetros de simulação:** Diferentemente dos cenários simulados com modelos de movimento sintéticos, os *traces* de movimentos apresentam uma densidade de nós variável dentro de uma área limitada e uma distribuição de velocidades não uniforme (ver Figura (3.6)). Para determinar parâmetros similares nos cenários com modelos de movimento sintéticos, utilizou-se a média amostral do número de nós no cenário de movimento dos *traces*, obtendo-se um valor de 254 nós. Para manipular a distribuição da velocidade dos movimentos sintéticos, que originalmente seria uniforme entre a faixa de velocidades mínima e máxima, criaram-se quatro grupos de nós com diferentes faixas de velocidades mínimas e máximas. Cada grupo inclui a uma porcentagem do número total de 254 nós. O número de nós e os diferentes valores de faixas de velocidades mínimas e máximas, atribuídos a cada grupo, permite obter uma distribuição de velocidades similar à dos *traces* de movimento. Por motivos de granularidade, isto é, por usar só quatro grupos de nós com faixas de velocidades diferentes, não se conseguiram distribuições de velocidades exatamente iguais, porém, sim similares. Na Tabela (3.1) mostra-se os parâmetros de configuração para os diferentes cenários. As configurações são efetivas para este capítulo, como para experimentos posteriores.

**Algoritmos de roteamento avaliados:** O objetivo de caracterizar e diferenciar os diferentes tipos de rede é a de encontrar alguma relação entre os padrões encontrados




(a) (b)

**Figura 3.6: (a) CDF das velocidades dos três modelos de movimento. (b) número de nós dos *traces* nas 5 horas de simulação, desde as 7 horas da manhã (maior quantidade de nós) até as 12 horas.**

com o desempenho obtido dos algoritmos de roteamento avaliados e segundo o funcionamento de cada um deles. Os algoritmos considerados são os algoritmos de roteamento de múltiplas cópias: *Spray and Wait*(SPYROPOULOS; PSOUNIS; RAGHAVENDRA, 2005), referenciado como *SaW*, *Prophet*(LINDGREN; DORIA; SCHELÉN, 2003) e *Epidemic*(VAHDAT; BECKER, 2000), explicados anteriormente na Seção (2.2). Para o algoritmo *SaW* realizaram-se seis simulações para cada modelo de movimento com seis valores diferentes de número do número de contagem inicial de cópias $L$, onde $L = \{5, 10, 30, 50, 100, 150\}$ (ver Seção (2.2.1.2)). Para o protocolo *Prophet* também realizaram-se seis simulações com diferentes valores de tempo de envelhecimento $t$, onde $t = \{60, 180, 240, 300, 600, 1200\}$ (ver Seção (2.2.2.1)). Na seção de avaliação dos resultados (3.4.3), apresentam-se os resultados de cada algoritmo. No caso dos algoritmos *SaW* e *Prophet*, apresentam-se os resultados do cenário com o parâmetro que obteve o maior desempenho para cada modelo de movimento. No caso do algoritmo *SaW*, os melhores resultados se deram com o valor maior de $L = 150$, para todos os modelos de movimento. No caso de protocolo *Prophet*, a diferencia entre os resultados com diferentes valores de $t$, não é significativa. Para este algoritmo escolheram-se os resultados do cenário com um $t = 300$. Para o protocolo *Epidemic* não se configurou nenhum parâmetro especial ou medida contra a inundação de cópias de mensagens.

**Tabela 3.1: Parâmetros usados para a simulação de cenários.**

| Modelo de movimento | *Traces* | Modelos sintéticos (*RWP* e *SPMB*) |
|---|---|---|
| Taxa de transmissão | 5mbps | 5mbps |
| Rádio de transmissão | 250m. | 250m. |
| Espaço de armazenamento | Ilimitado | Ilimitado |
| Tamanho dos pacotes | 128KB. | 128KB. |
| Intervalo de criação de mensagens | aprox. 60s. | aprox. 60s. |
| TTL | sem | sem |
| Nro. de grupos de nós | 1 | 4 |
| Nro. de nos | Variável | Total de 254 nós.<br>Grupo1:71 nós<br>Grupo2:99 nós<br>Grupo3:61 nós<br>Grupo4:23 nós |
| Faixas de velocidades | 0 a 22 m/s | Por grupos:<br>Grupo1:0 a 2 m/s<br>Grupo2:2 a 5 m/s<br>Grupo3:5 a 10 m/s<br>Grupo4:10 a 22 m/s |

### 3.4.2 Caracterização das topologias de rede dos modelos de movimento estudados

Avaliam-se três tópicos: a conectividade entre pares, a conectividade da rede e, a qualidade dos nós e a sua relação com a zona geográfica.

**Estudo da conectividade entre pares:** Na Figura (3.7) são mostrados os resultados das métricas para avaliar a conectividade entre pares de nós. O conjunto de diagramas revela que em cenários nos quais os nós têm maiores restrições de movimento, apresentam-se melhores condições de conectividade entre pares para a transmissão de dados. Deste modo, segundo estes resultados, os cenários mais propícios para a transmissão de dados entre pares são (em ordem): (i) os *traces* de movimento, (ii) o cenário *SPMB* e, por último (iii) *RWP*. No diagrama (3.7a), os *traces* de movimento apresentam tempos de contato mais elevados, enquanto *RWP* tempos de contato baixos. No diagrama (3.7b), o cenário *SPMB* apresenta os tempos entre contatos mais baixos quando comparado com os outros modelos de movimento. Isto implica que no cenário *SPMB*, algoritmos de roteamento como *Prophet*, podem um obter um bom desempenho em probabilidade de entrega, quando comparado com

outros algoritmos, uma vez que *Prophet* realiza repasse de cópias de mensagens segundo a probabilidade de encontros dos nós com o nó destino. Os resultados do diagrama (3.7b) se complementam com os resultados do diagrama (3.7d). O diagrama (3.7d) mostra o número de mudanças do estado dos enlaces entre pares de nós. Esta informação revela, uma vez mais, que os algoritmos de roteamento que usem como métrica a probabilidade de encontros, podem obter bom desempenho no cenário *SPMB*. Isto porque, pares de nós neste cenário se encontram mais vezes do que os nós nos outros dois cenários. Nos cenários *RWP* e dos *traces*, o diagrama (3.7d) revela que pares de nós não se encontram frequentemente e, as faculdades dos algoritmos baseados na probabilidade de encontros, não poderiam ser bem aproveitadas. Finalmente, no diagrama (3.7c), apresentam-se os resultados para cada cenário da duração dos enlaces entre pares de nós considerando as mudanças do estado do enlace. Os resultados revelam que os cenários com maiores restrições para o movimento dos nós apresentam maior duração do enlace, deste modo, os cenário dos *traces* e do *SPMB* possuem maior estabilidade de enlaces entre pares do que o cenário *RWP*.

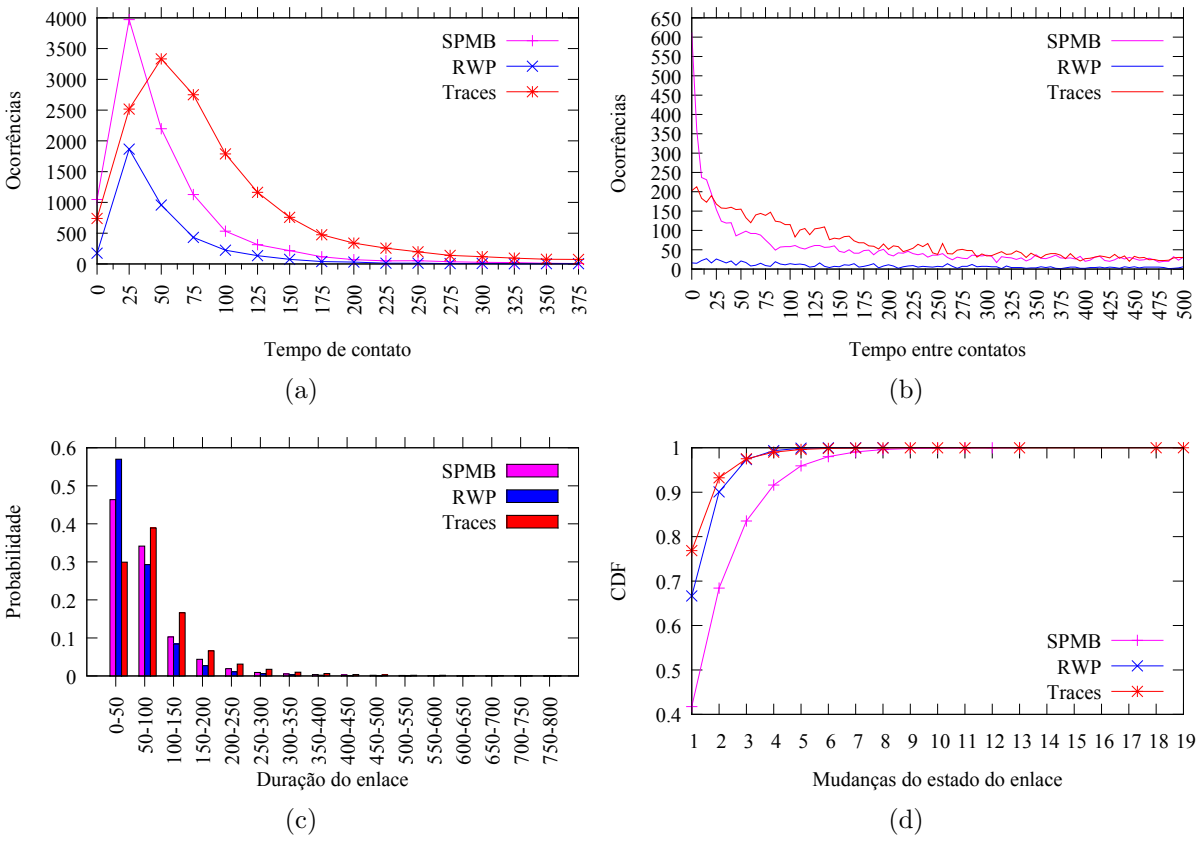
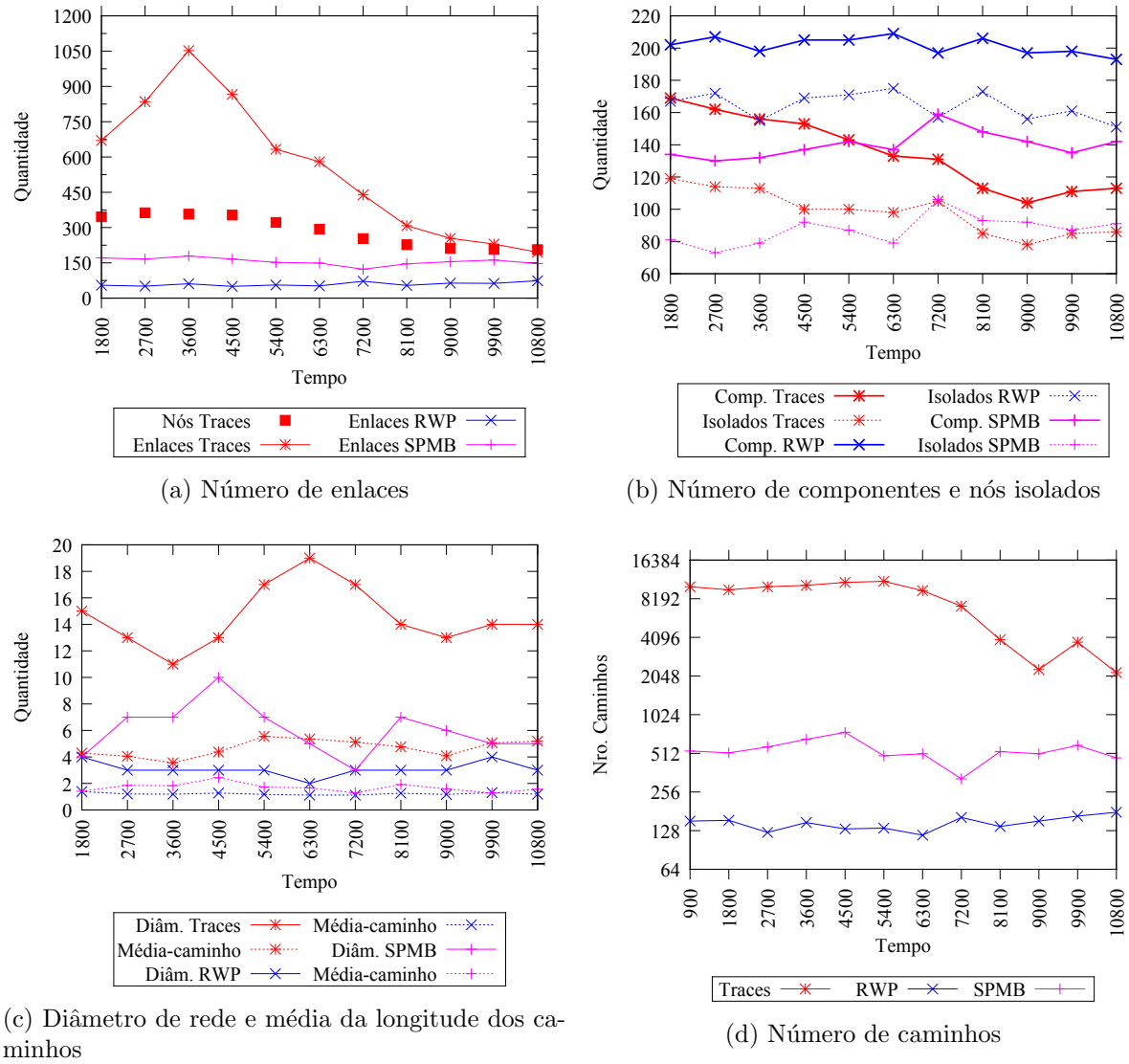



**Figura 3.7: Propriedades da conectividade entre pares.**

**Caracterização da conectividade de rede:** Nos *traces* de movimentos o número de nós que permanecem dentro da área de estudo é variável e com uma média de 254 nós. A partir da segunda hora de simulação o número de nós diminui e permanece em valores menores do que a média. Os cenários com movimento sintético utilizam um número fixo de nós igual a 254. Não obstante, o número de enlaces no grafo de conectividade dos *traces* é sempre maior ao número de enlaces dos grafos de conectividade nos cenários com os outros modelos de movimento (3.8a). Isto determina que não é suficiente considerar só o número total de nós em uma rede para o estudo de sobrecarda de rede. Em termos de conectividade, o pior cenário é o cenário do movimento *RWP*, com baixa quantidade de enlaces, maior quantidade de nós isolados e componentes de rede. A rede do cenário *RWP* é uma rede altamente fragmentada com baixa quantidade de rotas de conectividade. Esta rede pode obter baixo desempenho utilizando algoritmos de roteamento de inundação controlada. Entretanto, é um cenário em que o algoritmo *Epidemic* poderia melhorar o desempenho consideravelmente. Depois da segunda hora de simulação, momento em que o cenário dos *traces* apresenta similar o menor quantidade de nós do que o cenário *SPMB*, o digrama (3.8b)) mostra que ambos os cenários apresentam similar quantidade de nós isolados (componentes de um só nó). Não obstante, o cenário *SPMB* apresenta maior quantidade de componentes de rede. Isto significa que o grafo de conectividade do cenário *SPMB* apresenta componentes de menor tamanho do que os componentes do cenário dos *traces*. A diferencia entre os tamanhos dos componentes de rede, nos dois cenários, pode estimar-se segundo o diagrama (3.8c). No diagrama (3.8c) mostra-se o diâmetro da rede, que em caso de redes não totalmente conectadas, é possível que pertença ao diâmetro de uns dos componentes de maior tamanho. O modelo de movimento dos *traces* apresentam um diâmetro de rede maior, comparado com os outros modelos, e assim, apresenta componentes de tamanho grande.

Finalmente, o gráfico (3.8d) mostra a relação entre o número de caminhos mais curtos que existem nas diferentes topologias de rede criadas pelos diferentes modelos de movimento, a quantidade, no caso dos *traces*, supera consideravelmente às outras topologias de rede. O impacto na congestão de rede por parte dos algoritmos de elevado e muito superior, quando comparado com os cenários dos outros modelos de movimento. Contudo, existe a dificuldade de que ainda com a existência de componentes altamente conectados no cenário dos *traces* de movimento, que representa o movimento dos veículos nas *VANETs*, existem componentes isolados e fragmentação

da rede. Neste cenário, os algoritmos de roteamento como *Epidemic* criariam muita sobrecarga de rede. Não obstante, os componentes de rede altamente conectados absorveriam a disseminação de cópias de mensagens de algoritmos que controlem a inundação, como o *SaW*. Precisa-se de abordagens que ajam segundo as condições do ambiente no momento da transmissão e segundo a densidade de diferentes zonas.




(a) Número de enlaces

(b) Número de componentes e nós isolados

(c) Diâmetro de rede e média da longitude dos caminhos

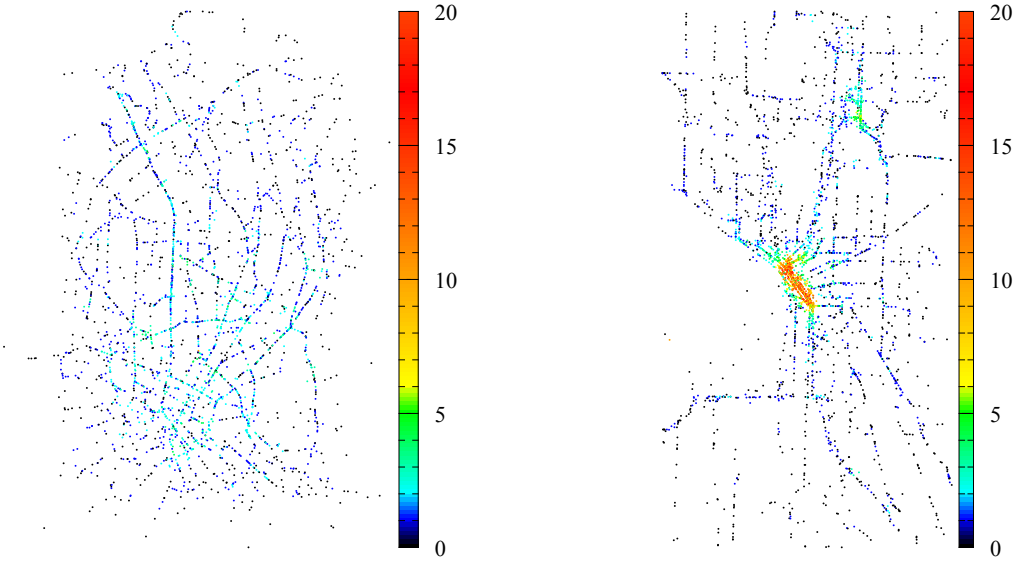
(d) Número de caminhos

**Figura 3.8: Propriedades da conectividade na topologia de rede.**

**Qualidade dos nós e a sua relação com a zona geográfica:** Os nós de maior qualidade são aqueles que podem repassar mensagens a vários nós. Segundo (PALLIS et al., 2009) pode-se afirmar o seguinte sobre qualificação de nós:

- O grau de conectividade, ou número de vizinhos, não determina a qualidade dos nós.
- A métrica de centralidade *Betweenness* pode expressar qualidade de um nó tanto quanto a métrica de *lobby index*.
- A métrica de *lobby index* (KORN; SCHUBERT; TELCS, 2009) está correlacionada à densidade de nós. Obtendo uma correlação de *Pearson* de 0.711 no cruzamentos das ruas.

Pelo concluído, segundo (PALLIS et al., 2009), o valor de *lobby index* pode ser utilizado tanto para qualificar nós, como para detectar a densidade de nós em uma zona. Assim, criou-se um relatório no simulador para obter os valores de *lobby index* segundo a localização dos nós para os mapas dos cenários *SPMB* e dos *traces*. Na Figura (3.9) mostra-se o resultado. Nos resultados, o mapa do modelo de movimento sintético apresenta uma distribuição dos valores altos de *lobby índex* mais dispersa. Diferentemente, o mapa dos *traces* mostra claramente duas zonas de maior densidade de veículos.



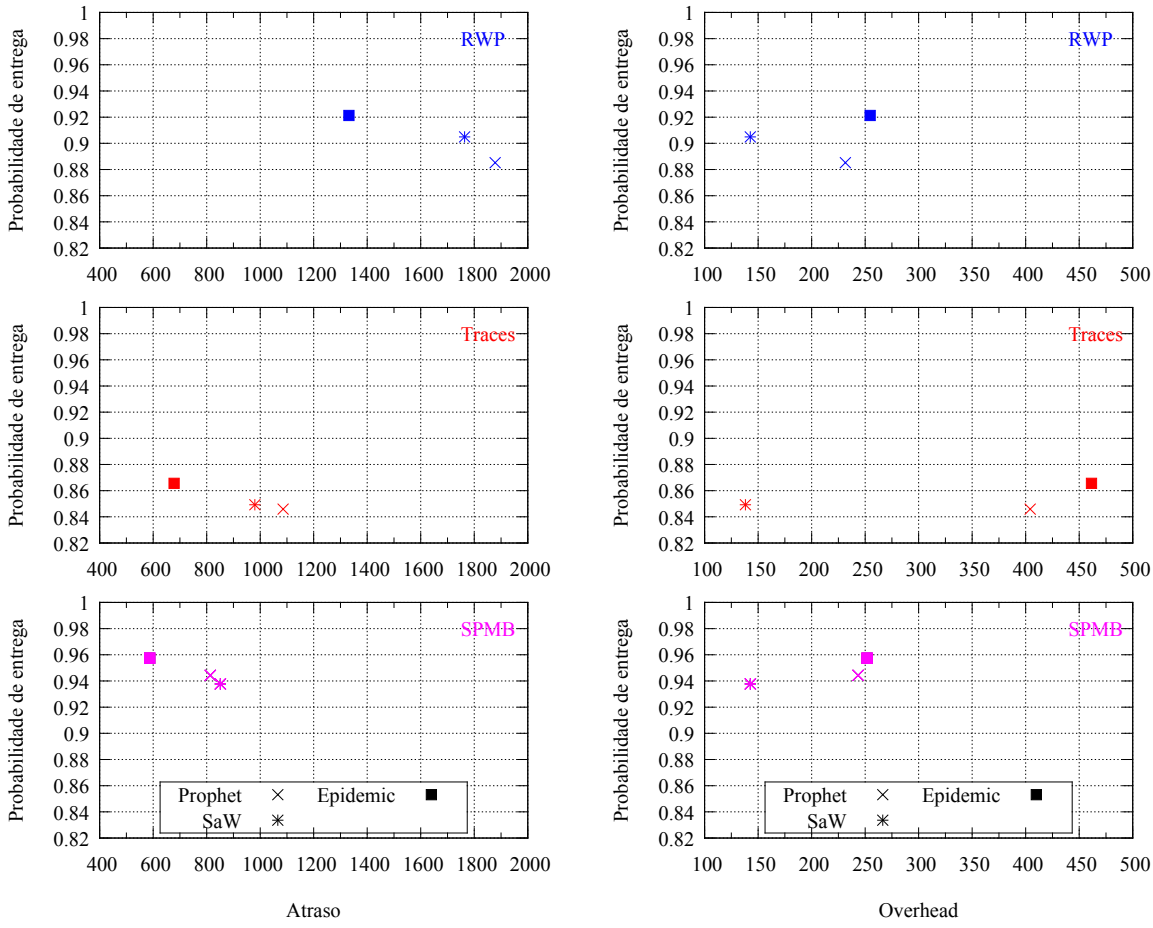

(a) SPMB (Manchester) (b) *Traces* de movimento

**Figura 3.9: Relação entre *lobby index* e as localizações dos nós.**

### 3.4.3 Avaliação dos resultados de desempenho

Nesta seção apresentam-se os resultados de desempenho dos algoritmos de roteamento avaliados: (i) *Prophet*, com tempo de envelhecimento de $t = 300$, (ii) *SaW*, com número de

contagem de cópias inicial de $L = 150$ e, (iii) *Epidemic*, sem nenhum parâmetro de controle de inundação configurado.




(a) Prob. Entrega Vs. atraso

(b) Prob. Entrega Vs. overhead

**Figura 3.10: Probabilidade de entrega dos algoritmos *Epidemic*, *SaW* e *Prophet* aplicados as modelos de movimento estudados.**

Na Figura (3.10) são mostrados os resultados da probabilidade de entrega de mensagens e a relação entre a probabilidade de entrega com o atraso e a sobrecarga ou *overhead* de rede. O *overhead* é a relação entre as transmissões adicionais de cópias de mensagens que se transmitiram para entregar os mensagens e o número de mensagens entregues ao destino. O *overhead* calcula-se segundo a Equação (3.1).

$$\frac{Nro.\ de\ transmissões - Nro.\ de\ mensagens\ entregues}{Nro.\ de\ mensagens\ entregues} \tag{3.1}$$

Segundo as propriedades conhecidas do algoritmo *Epidemic*, era previsto que obtenha a maior probabilidade de entrega, o menor atraso e, a maior sobrecarga de rede em todos os

cenários, quando comparado com os outros algoritmos. No entanto, no cenário dos *traces* a sobrecarga de rede é de quase o dobre da sobrecarga nos outros cenários. Isto é porque o cenário dos *traces* corresponde a uma rede com componentes de rede de grande tamanho e altamente conectados. Desta forma, ainda em uma rede de moderada densidade, os componentes de rede altamente conectados criam momentos de alta sobrecarga de rede. No caso do algoritmo *Prophet*, a sobrecarga de rede é próxima à sobrecarga gerada pelo algoritmo *Epidemic*. O algoritmo cuja sobrecarga de rede é independente da conectividade entre nós, é o algoritmo *SaW* por ter um número de cópias fixo a disseminar.

O cenário com movimento *SPMB*, foi determinado previamente como um bom cenário para a aplicação do algoritmo de roteamento *Prophet*, uma vez que às suas características de mudanças do estado enlace (3.7b) e tempos entre contatos (3.7d), determinam que os nós encontram-se em tempos pequenos e com maior frequência que em os outros cenários de movimento. Os resultados confirmam o previsto, o cenário com movimento *SPMB*, é o único caso em que o algoritmo *Prophet* supera o em probabilidade de entrega e atraso ao algoritmo *SaW*. Nos outros dois cenário o algoritmo *Prophet* têm baixo desempenho quando comparado com o algoritmo *SaW*.

Contudo, comparando o efeito do movimento no desempenho dos algoritmos. Os movimentos sintéticos, cujos movimentos dos nós são livres ou com menores restrições, favorecem consideravelmente ao algoritmo *Epidemic* na avaliação da média de atraso sem considerar a sobrecarga real que causaria pela restrição do movimento dos nós, as quais, determinam zonas de nós densas que influenciam consideravelmente na sobrecarga de rede. Assim, considerar só o número total de nós em uma rede, sem considerar a distribuição dos nós no espaço com o decorrer do tempo, não representa cenários reais e adequados para a avaliação do desempenho da rede, principalmente na avaliação da sobrecarga de rede gerada pela disseminação de cópias.

# Capítulo 4

## Roteamento em VANETs usando algoritmos de classificação

O comportamento das *VANETs* em ambientes urbanos está influenciado pela demografia de uma cidade e por fatores como: sinais de tráfego e os padrões de tráfego veicular que dependem da hora do dia. Estes fatores impõem padrões de movimento nos veículos e determinam zonas geográficas de maior densidade. A disseminação e o roteamento de mensagens estão altamente influenciados pela densidade de veículos em uma determinada zona. Assim, alguns algoritmos foram projetados para agir diferentemente baixo condições extremas de baixa e alta densidade (TONGUZ et al., 2007).

Este capítulo descreve como algoritmos de aprendizado de máquina, especificamente classificadores, podem ser aplicados a algoritmos de roteamento de múltiplas cópias para aprimorar o seu desempenho em *VANETs*. Os algoritmos, cujo comportamento foi modificado, são o algoritmo *SaW* em modo binário (SPYROPOULOS; PSOUNIS; RAGHAVENDRA, 2005) e o algoritmo *Epidemic*(VAHDAT; BECKER, 2000).

Estuda-se particularmente o efeito dos algoritmos projetados em *VANETs* formadas por ônibus de transporte público porque apresentam comportamentos prediziéis e periódicos. As características de movimento e conectividade, dos *traces* de movimento utilizados para a simulação dos cenários, estão descritas no Capítulo (3) e nas seções:(3.3.1) e (3.4.2).

Este trabalho foi inspirado na utilização do algoritmo de classificação *Naive Bayes* em trabalhos como (COLAGROSSO, 2005) e (AHMED; KANHERE, 2010). Os algoritmos propostos nesses trabalhos usaram esse classificador para a aprendizagem das condições baixo as quais acontecem certos eventos que podem influenciar o desempenho. Os algoritmos de disseminação de *broadcast* (COLAGROSSO, 2005) e de roteamento por cópia simples (AH-

MED; KANHERE, 2010), podem predizer que eventos poderiam acontecer e agir segundo o previsto. Em (AHMED; KANHERE, 2010) utilizam-se os mesmos *traces* de movimento, porém, aplicado a uma área diferente do mapa geográfico e com uma baixa densidade de veículos (35 veículos). Não obstante, nosso estudo baseia-se na avaliação de algoritmos de roteamento de múltiplas cópias e, para a avaliação da sobrecarga de rede, considerou-se necessário manter uma densidade alta correspondente a uma média de 254 nós.

## 4.1 Considerações

Projetaram-se algoritmos de roteamento com suporte de classificadores para aprimorar o desempenho dos algoritmos: (i) *SaW* e, (ii) *Epidemic*.

Para nossa abordagem, considera-se necessário coletar informação da zona onde os nós estiveram localizados em determinados eventos. Para isto, cada veículo há de contar com uma representação do mapa geográfico da cidade, ou da área de movimentação, dividida em quadrados de $1km^2$. Codificou-se cada zona com um identificador que os nós informam como característica da sua localização. A influência das zonas de uma cidade segundo a sua densidade é diferente segundo o funcionamento do algoritmo. No caso do algoritmo *Epidemic* a correspondência entre a densidade de nós dos componentes da rede é clara; zonas de maior densidade causam maior sobrecarga de rede e, em zonas de baixa densidade, precisa-se de uma maior propagação de cópias de mensagens. Por este motivo, nos experimentos da Seção (3.4) do Capítulo (3), o protocolo *Epidemic* apresenta menos sobrecarga de rede nos modelos de *RWP* e *SPMB* do que no cenário com *traces*. Isto se deve a topologia de rede gerada pelos os *traces* de movimento apresenta componentes de redes altamente conectados, caso contrário das redes dos outros modelos de movimento. Na avaliação do algoritmo *SaW* (ver (3.4)), utilizando um número de cópias de 150 no cenário dos *traces* de movimento. Nos relatórios observou-se no 80% dos casos as mensagens chegaram ao destino com um valor de cópia de uma unidade. Isto significa que em um 80% dos casos de entrega da mensagem, os nós ficaram com o número mínimo de cópias e, a partir de esse momento, o atraso do envio depende de que tão rápido os nós se movimentaram para encontrar ao destino (entrega por *direct transmisson*). Por causa disto é importante a qualidade do nó como retransmissor e a retransmissão dos nós em zonas de maior concentração, onde tenham maior probabilidade de encontrar a vários nós. As zonas de baixa concentração de nós poderiam prejudicar o desempenho do algoritmo *SaW* porque existe menos probabilidade de encontrar nós retransmissores e nós destinos. O impacto seria maior se um nó realiza-se os primeiros repasses de uma mensagem (com

os valores maiores de número de cópias) a nós em uma zona de baixa densidade. Em razão do exposto, sobre os dois algoritmos, considera-se o seguinte:

- O algoritmo de roteamento *Epidemic* causa alta sobrecarga de rede quando os nós retransmitem em zonas de alta densidade de nós e grau de conectividade. Não obstante, em zonas de baixa densidade o comportamento padrão do algoritmo permite maior probabilidade de entrega e menor atraso.
- O algoritmo de roteamento *SaW* o repasse a nós em zonas de baixa densidade poderiam incrementar o atraso do envio das mensagens até o destino, ou seja, os nós em zonas de baixa densidade poderiam demorar em encontrar outros nós

O processo de classificação, posteriormente descrito, foi realizado em base às duas considerações expostas anteriormente. Adicionalmente pensa-se que existe uma relação entre a densidade da zona e o tempo que demora um nó em retransmitir uma mensagem. Em zonas de maior densidade, os nós encontram maior quantidade de possíveis retransmissores e tendem a transmitir num tempo pequeno, caso contrário em zonas de baixa densidade. Dependendo da granularidade, na divisão do mapa de uma cidade, as zonas podem incluir várias rotas. Assim, considera-se que alguns nós podem retransmitir em tempo menor do que outros na mesma zona, dependendo da rota seguida. Por este motivo é necessária a coleta de informação adicional que possa distinguir aqueles nós dos outros. Especificam-se o conjunto de dados coletados na seção de coleta de tuplas anotadas (Seção (4.2.1)).

Finalmente, neste projeto utilizou-se o algoritmo *C4.5* de (QUINLAN, 1993), que utiliza uma árvore de decisão para extrair a lógica de classificação. Este algoritmo foi escolhido pelas referências do seu bom desempenho em precisão e tempo de execução (LIM; LOH; SHIH, 2000). Posteriormente, o algoritmo *C4.5* aplicado aos conjuntos de dados de treinamento deste projeto e, comparado com o algoritmo *Naive Bayes*, revelou maior precisão no processo de classificação.

## 4.2 Descrição do método

O procedimento para aproveitar o suporte do algoritmo de classificação *C4.5* consiste em três fases principais: (i) a coleta de tuplas anotadas, (ii) treinamento do classificador e, (iii) roteamento utilizando classificação. Este procedimento será aplicado em cada algoritmo de roteamento (*SaW* e *Epidemic*). Os algoritmos posteriormente projetados

modificaram a lógica de roteamento padrão segundo o suporte dos classificadores treinados para cada algoritmo de roteamento.

Na Figura (4.1) mostra-se o procedimento para aplicar classificadores no processo de roteamento de mensagens. A fase (i) consiste na execução do algoritmo de roteamento com o funcionamento padrão, coletando as tuplas anotadas para a aprendizagem do classificador nos momentos desejados. A fase (ii) consiste na leitura das tuplas anotadas e treinamento do classificador. O treinamento do classificador realiza-se segundo o algoritmo de classificação utilizado (*c4.5*). Na fase (iii) os nós descrevem a situação presente baixo as quais devem decidir se repassar a mensagem ou não a outro nó. Esta ação realiza-se utilizando um conjunto de atributos não anotados. Este conjunto de atributos é enviado ao classificador treinado para determinar o atributo de anotação de classe. Segundo o atributo de anotação de classe obtido e o algoritmo de roteamento utilizado, os nós repassam ou não a mensagem a outro retransmissor. Cada uma das fases mostradas na Figura (4.1) são explicadas nas seções posteriores.

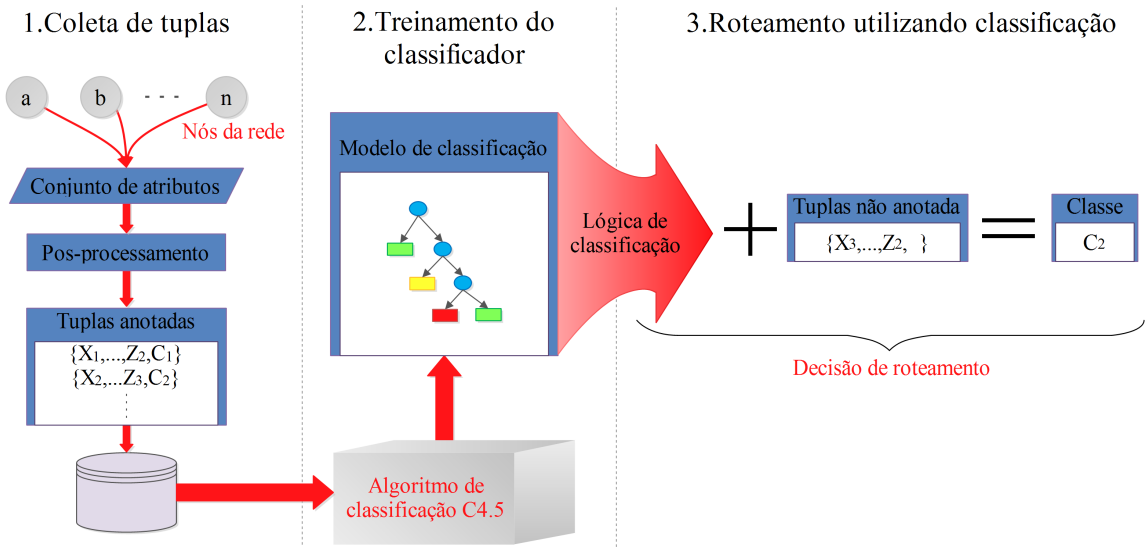



**Figura 4.1: Descrição do método para a aplicação do classificador no roteamento de mensagens.**

### 4.2.1 Coleta de atributos de tupla

Nas simulações a coleta de atributos realiza-se com um único objeto instanciado de uma classe de relatório desenvolvida. Este objeto cadastra os valores coletados em um único arquivo auxiliar para o posterior processamento. O resultado é um único arquivo com dados de treinamento para o classificador. Inicialmente, optou-se por realizar a coleta de atributos individualmente em cada nó para o treinamento de classificadores individuais,

porém, a quantidade e qualidade de dados coletado em cinco horas de simulação, não foi o suficiente. Não obstante, não se descarta a possibilidade deste tipo de implementação em cenários reais.

O processo de coleta inicia-se quando um nó começa a receber uma mensagem (inicio da transferência da mensagem). Neste momento são coletados alguns valores temporais que incluem atributos e valores para o cálculo de outros atributos. O processo de coleta pode ser interrompido quando a transferência da mensagem não teve sucesso (transmissão abortada), evento que pode acontecer quando um nó movimenta-se fora do raio de alcance do nó transmissor. Neste caso, os valores temporais coletados são excluídos e procede-se a esperar o seguinte evento de inicio de transferência de mensagem. Caso contrário, a coleta de atributos é concluída quando um nó, que recebeu a mensagem com sucesso, conseguiu retransmiti-la. No momento da retransmissão o nó coleta outros dados temporais e realiza os cálculos necessários para completar a coleta de atributos. Os atributos coletados são enviados ao objeto da classe do relatório para o seu cadastro em disco num arquivo auxiliar. Na Figura (4.2) mostra-se o procedimento realizado para a coleta de atributos.

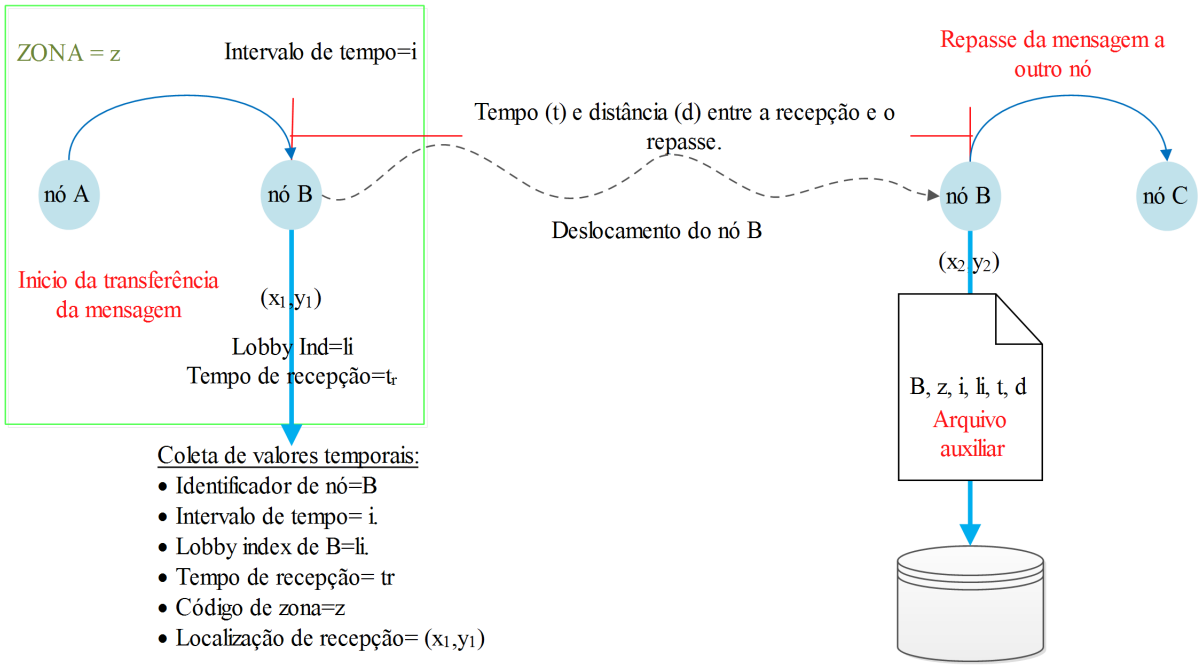



**Figura 4.2: Coleta de atributos e valores auxiliares para a classificação.**

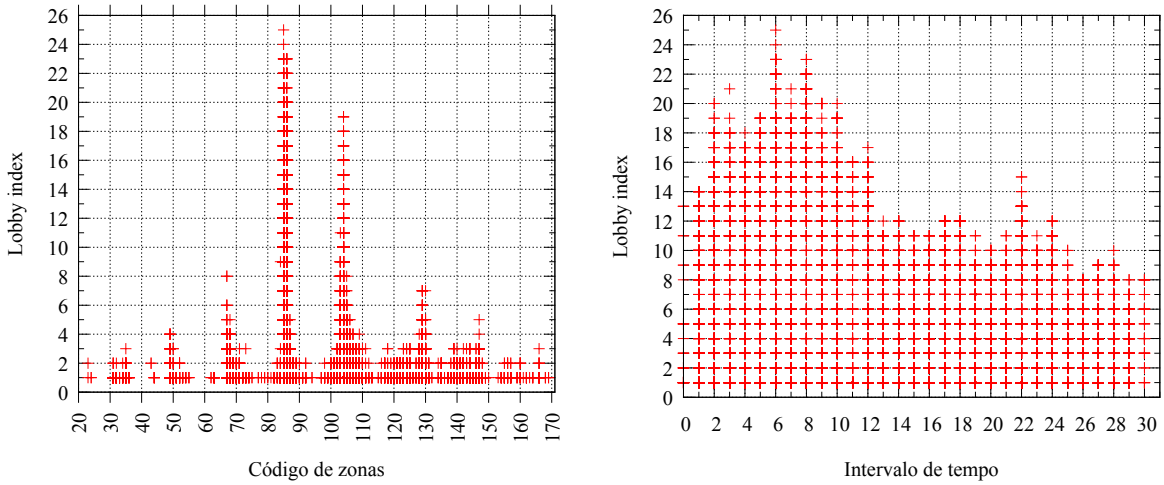
É necessário esclarecer que o arquivo auxiliar, gerado pela coleta de atributos, não contém os dados finais para o treinamento do classificador. Este arquivo será processado para obter finalmente o conjunto de dados de treinamento. Posteriormente, explicar-se-á como se realiza este processo. A continuação explicam-se os atributos coletados neste

arquivo auxiliar.

**Atributos coletados:** Dados um nó $y$ que retransmitiu uma mensagem recebida do nó $x$, os atributos coletados podem se descrever da seguinte maneira:

- Código do nó retransmissor: É o código do nó que recebeu a mensagem e conseguiu retransmiti-la ($y$). A intenção de incluir este atributo é de que quando um nó $x$ deseje transmitir uma mensagem ao nó $y$, consulte ao classificador treinado pelo atributo de classe que determina a qualidade de $y$ como retransmissor.
- Código da zona: É a zona da cidade onde esteve localizado o nó $y$ no momento em que recebeu a mensagem do nó $x$. Utiliza-se este atributo porque, segundo a característica da zona, a dificuldade para retransmitir uma mensagem pode ser maior o menor para um nó.
- Intervalo de tempo: Na Figura (3.6) mostrou-se que o tempo influencia na densidade de nós. Em vista de que a densidade de nós de uma zona pode variar com o decorrer do tempo, decidiu-se incluir este valor como atributo de tupla. Os intervalos de tempo são de 600 segundos e informam o intervalo de tempo em que o nó $y$ recebeu a mensagem de um nó $x$ para retransmiti-la.
- *Lobby index*: O valor de *lobby index* é utilizado para descrever a densidade de nós que percebe o nó $y$ quando recebeu uma mensagem. Para obter o valor de *Lobby index* cada nó poderia enviar o seu número de vizinhos em um salto (*one-hop*) em mensagens de *beacons* a cada vizinho. *Beaconing* é um mecanismo em que os nós realizam *broadcasts*, com certa frequência, para enviar mensagens do seu estado a seus vizinhos mais próximos (*one-hop*). Embora muitas abordagens considerem a troca de informação utilizando *beaconing* (SOMMER; TONGUZ; DRESSLER, 2010) (WISCHHOF; EBNER; ROHLING, 2005) (TONGUZ; WISITPONGPHAN; BAI, 2010), seu uso é um tópico ainda questionável pelos problemas no desempenho poderiam ocasionar. O impacto da utilização do *beaconing* é um tema ainda pesquisado (EENENNAAM et al., 2009),(WOLTERINK; HEIJENK; KARAGIANNIS, 2011). Segundo (EENENNAAM et al., 2009), dependendo da carga da rede, poderia se utilizar um canal exclusivo para o *beaconing* e outro para a transmissão de mensagens da dados.

  Na Figura (4.3) é mostrada a correlação encontrada, nas tuplas coletadas, entre as zonas e o intervalo de tempo com o valor de *lobby index*. No diagrama (4.3a)



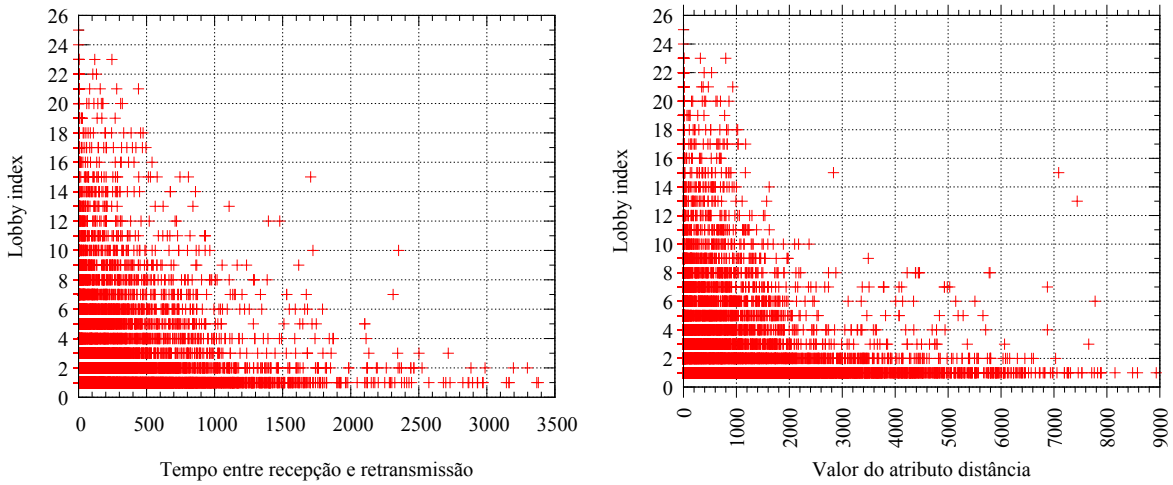

(a) Relação entre o *lobby index* e as zonas. (b) Relação entre o *lobby index* e os Int. de tempo.

**Figura 4.3: Relação entre o atributo *lobby index* com os atributos de zona e intervalo de tempo.**

aprecia-se que algumas zonas apresentam valores de *lobby index* altos e outras baixos, o que implica maior e menor congestão de veículos em algumas zonas. O diagrama (4.3b) mostra que o valor do *lobby index* diminui com o tempo. Isto concorda com a tendência da densidade total da rede que diminui com o decorrer do tempo, segundo a Figura (3.6).

- Tempo entre recepção e retransmissão: Este atributo determina o tempo que demorou o nó $y$ em retransmitir a mensagem recebida do nó $x$. Escolheu-se este atributo pela suposição de que um nó pode retransmitir uma mensagem rapidamente em uma zona de maior densidade. Com a informação da coleta de tuplas, comprovou-se que existe essa correlação. No digrama (4.4a) mostra-se que o tempo que um nó demora em retransmitir uma mensagem incrementa-se a menor valor de *lobby index*; ou seja, com menor de densidade de nós.

- Distância entre as localizações de recepção e retransmissão: Este atributo determina a distância entre o ponto de localização onde o nó $y$ recebeu a mensagem e o ponto de localização onde $y$ a retransmitiu. O conjunto de dados da coleta de atributos revelou que existe a correlação entre este atributo e o valor de *lobby index*. No digrama (4.4b) mostra-se o valor de este atributo é menor a medida que o valor de *lobby index* é maior. Isto indica que quando um nó receber uma mensagem em uma zona de alta densidade encontra possíveis nós retransmissores a uma distância pequena devido à alta densidade de nós da zona.




(a) Relação entre o *lobby index* e o tempo de retransmissão.

(b) Relação entre o *lobby index* e o atributo distância.

**Figura 4.4: Relação entre o atributo *lobby index* com os atributos de tempo entre recepção e retransmissão e distância.**

Destes seis atributos coletados no arquivo auxiliar os atributos: (i) código do nó retransmissor, (ii) código da zona, (iii) intervalo de tempo, (iv) *lobby índex*, (v) tempo entre recepção e retransmissão e, (vi) distância entre as localizações de recepção e retransmissão. Só os quatro primeiros serão atributos de tupla do conjunto de dados de treinamento. Os atributos (v) e (vi) são atributos usados para a atribuição do atributo de anotação de classe da tupla do conjunto de dados de treinamento. Para isto, processa-se o conjunto de dados do arquivo auxiliar. Esta tarefa descreve-se a continuação.

**Processamento do arquivo auxiliar de tuplas coletadas:** O atributo de anotação de classe é designado segundo a relação entre os valores dos atributos: (i) tempo entre recepção e retransmissão (t) e, (ii) distancia entre as localizações de recepção e retransmissão (d). O objetivo é classificar os nós dependendo do tempo e a distância em que retransmitem desde o momento e a localização em que receberam uma mensagem. Deste modo poder-se-ia distinguir nós que retransmitem em pouco tempo e a uma maior distância de aqueles que retransmitiram a uma distância pequena em um tempo elevado. A distribuição dos valores de $\frac{d}{t}$ permite criar $m$ grupos de tuplas com valores próximos de $\frac{d}{t}$, que determinaram um conjunto de atributos de anotação de classes $C = \{C_1, \cdots, C_m\}$. Deste modo, o número de atributos de anotação de classe dependerá da distribuição de valores de $\frac{d}{t}$ e do número de tuplas coletadas. O objetivo é obter uma distribuição de valores de atributos de anotação de classe, do conjunto de dados de treinamento, quase uniforme para melhor preci-

são do classificador. A seguinte expressão representa a conversão de uma tupla do arquivo auxiliar a uma tupla anotada do arquivo de treinamento do classificador:

$$\begin{array}{c} \textit{Tupla do arquivo auxiliar}: \\ \left\{ idNó_B \quad z_4 \quad i_1 \quad li_4 \quad t_5 \quad d_3 \right\} \end{array} \xrightarrow{\frac{d_i}{t_j} \textit{ relacionado a } C_2} \begin{array}{c} \textit{Tupla do arquivo de treinamento}: \\ \left\{ idNó_B \quad z_4 \quad i_1 \quad li_4 \quad C_2 \right\} \end{array} \tag{4.1}$$

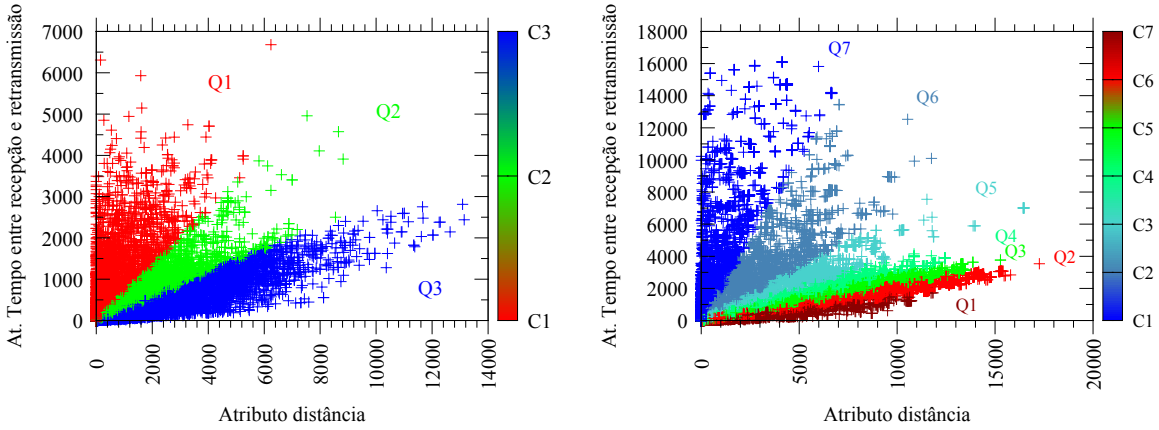
**Relacionamento entre atributos de anotação de classe e algoritmos de roteamento:** Os atributos foram coletados com o objetivo de caracterizar a densidade de nós de uma zona, em um determinado tempo e quando um nó segue uma determinada rota. O atributo de anotação de classe descreve o efeito causado na retransmissão de uma mensagem baixo as condições do ambiente e de rede. Isto é, um nó retransmite a uma determinada distância $d$ depois de um determinado tempo t. Adicionalmente, dado dos atributos de anotação $C_i$ e $C_{i+k}$, onde $k > 0$ temos que:

$$Se \ \frac{d}{t} \rightarrow C_i \ \ e \ \ \frac{d'}{t'} \rightarrow C_{i+k} \ \ se \ cumpre \ que \ \ \frac{d}{t} < \frac{d'}{t'} \tag{4.2}$$

Em razão do diferente proceder dos algoritmos de roteamento, o atributo de anotação de classe tem diferente significado para o algoritmo *SaW* e *Epidemic*. Na Figura (4.5) mostra-se relação entre os atributos de anotação de classes com o valores de $d$ e $t$ dos conjuntos de dados do algoritmo *SaW* e do algoritmo *Epidemic*. A coleta realizou-se executando os algoritmos com o seu funcionamento padrão, obtendo um total de três classes para o algoritmo *SaW* e sete para *Epidemic*. Na Figura (4.5) mostra-se também a relação entre as anotações de classe $C_i$, que cumprem com a expressão (4.2), e a situação favorável de retransmissão para cada algoritmo de roteamento, determinada por: $Q_j$, onde $j \in \mathbb{Z}^+$ e, a maior valor de $j$, a condição para retransmitir é mais favorável.

### 4.2.2 Treinamento do classificador

Neste projeto de mestrado utilizou-se um algoritmo de classificação, baseado em árvores de decisão, denominado *C4.5*. Este algoritmo utiliza as tuplas anotadas do conjunto de dados de treinamento para gerar uma árvore de decisão que representa a lógica de classificação. Desta maneira, um classificador treinado usa a árvore de decisão para classificar tuplas não anotadas. O funcionamento do algoritmo foi explicado anteriormente na Seção




(a) *SaW*. (b) *Epidemic*.

**Figura 4.5: Atributo de anotação de classe segundo os valores da distância e tempo entre recepção e retransmissão. Relação entre os atributos de anotação de classe e seu significado para cada algoritmo de roteamento(valor de $Q$).**

(2.2.3.1).

Para incluir o processo de classificação no simulador *The ONE*, utilizou-se a *API* do *software WEKA*[1]. O *WEKA*(HALL et al., 2009) é um *software* livre escrito em *Java* que contem vários algoritmos de aprendizagem de máquina, incluindo os classificadores *Naive Bayes* e *C4.5*. O *software WEKA* pode se utilizar também direitamente por uma *GUI*, fornecendo direitamente os arquivos com extensão *arff*. O *GUI* do *WEKA* foi utilizado para comparar a precisão dos classificadores *Naive Bayes* e *C4.5* treinados com o conjunto de dados de treinamento obtidos na fase de coleta de atributos. Utilizando validação cruzada com dez partições, a precisão do algoritmo *Naive Bayes* para o conjunto de dados de treinamento do algoritmo *SaW* foi de 58.46%, enquanto que para o algoritmo *Epidemic* foi de 47.08%. Os resultados para o algoritmo *C4.5* foram melhores; o algoritmo obteve 69.05% de precisão para o conjunto de dados do algoritmo *SaW* e 78.72% para o algoritmo *Epidemic*.

### 4.2.3 Roteamento usando classificação

Segundo a Figura (4.5), a relação entre os algoritmo de roteamento e os valores de anotação de classe $C_i$, pode se resumir da seguinte maneira:

**Algoritmo *SaW*:** Considerou-se que os melhores nós retransmissores são aqueles que

[1]http://www.cs.waikato.ac.nz/ml/weka/

retransmitem em pouco tempo e à maior distância possível. Assim, tem-se que:

- $C_1$: São nós que o tempo que demoram em retransmitir, não compensa a distância à que transmitem. Uma vez que estes nós podem retrasar o processo de retransmissão, decidiu-se excluí-los como possíveis retransmissores.
- $C_2$ e $C_3$: Os nós com $C_2$ são nós de qualidade média, os nós de $C_3$ são de boa qualidade. Em razão de que o algoritmo $SaW$ já é um algoritmo de controle de inundação de cópias, decidiu incluir aos nós de ambas as qualidades, no processo de retransmissão.

A continuação mostra-se a lógica de roteamento do algoritmo $SaW$ com classificação ($SaW$+$C$) (4.1):

**Algoritmo 4.1 Algoritmo *SaW+C* para retransmitir mensagens ao estabelecer uma conexão.**

**Entrada:** Nova conexão $x$, coleção de mensagens a retransmitir $M$
**Saída:** Ação $a \in A = \{$ iniciar retransmissão, não retransmitir $\}$

1: $n \leftarrow$ **outroNó**($x$)
2: **se** **éNóDestino**($n$) **então**
3: **retorna** iniciar retransmissão $M(n)$
4: **fim se**
5: $M' \leftarrow$ **obterMsNsComNroCopiaMaiorAum**($M$)
6: **se** $M'$ não tiver elementos **então**
7: **retorna** não retransmitir
8: **fim se**
9: $C_i \leftarrow$ **solicitarAtributoAnotação**($n$)
10: **se** $C_i$ é $C_1$ **então**
11: **retorna** não retransmitir
12: **senão**
13: **para** cada mensagem $m_i \in M' = \{m_1, m_2, \cdots, m_k\}$ **faça**
14: **se** nóAceitaMensagem($n, m_i$) **então**
15: **retorna** iniciar retransmissão
16: **fim se**
17: **fim para**
18: **retorna** não retransmitir
19: **fim se**

Dado um nó $x$, executando o algoritmo $SaW$+$C$, que estabeleça conexão com outro nó $y$, o funcionamento do algoritmo é o seguinte: se $x$ tiver uma mensagem com destino ao nó $y$, o nó $x$ decide iniciar a transmissão da mensagem (linhas 2 a 4). Em $SaW$+$C$ as mensagens com valor de cópias de replicação de uma unidade, devem entregar-se neste momento se houver conexão com o nó destino (*direct transmission*). Caso contrário, $x$ verifica que mensagens ainda possuem um valor de cópia

de replicação maior a uma unidade (linha 5). Se não tiver mensagens a replicar, o nó $x$ não inicia nenhuma retransmissão e o processo acaba até a próxima oportunidade de retransmissão. Se o nó $x$ apresentar mensagens que podem ser replicadas a nós retransmissores, solicita ao nó $y$ o atributo de anotação de classe dele (linha 9). Neste momento $y$ descreve a situação na qual transmitiria usando o conjunto de atributos. Esta tupla não anotada é enviada ao classificador treinado do nó $y$ que determina o atributo de anotação de classe. Quando o nó $x$ receber a classe de anotação do nó $y$, se o valor for $C_1$, decidi não retransmitir uma vez que as condições nas quais retransmitiria o nó $y$, fazem dele um nó de baixa qualidade como retransmissor (linhas 10 a 12). Se o valor de anotação de classe for outro, o nó $x$ tenta de enviar uma das mensagens que podem se replicar, ao nó $y$. A retransmissão é iniciada quando o nó $y$ aceita alguma das mensagens de $x$ (linha 14 a 16). A mensagem poderia ser rejeitada por o nó $y$, se o nó já possuísse a mensagem.

**Algoritmo *Epidemic*:** Considerou-se aqueles que retransmitem em pouco tempo são aqueles que ocasionam alta sobrecarga. Para o algoritmo *Epidemic* utilizou-se probabilidades de retransmissão fixas, segundo o valor da anotação de classe de um possível nó retransmissor. Como se mostra no diagrama (4.5b), dada um atributo de anotação de classe $C_i$, a maior valor de $i$ menor probabilidade de escolher um nó como retransmissor. Adicionalmente, quando um nó obtiver o atributo de anotação de classe $C_7$, exclui-se totalmente do processo de retransmissão. Na Tabela (4.1) mostrados os valores de probabilidade de entrega para cada atributo de anotação de classe:

**Tabela 4.1: Valores de probabilidade de retransmissão para cada atributo de anotação de classe $C_i$ no algoritmo *Epidemic+C*.**

| $C_i$ | Probabilidade de retransmissão ($p = [0,1]$) |
|---|---|
| $C_1$ e $C_2$ | Situações com dificuldade de retransmissão. Retransmitir sempre ($p = 1$) |
| $C_3$ | Retransmitir com probabilidade ($p = 0.80$) |
| $C_4$ | Retransmitir com probabilidade ($p = 0.50$) |
| $C_5$ | Retransmitir com probabilidade ($p = 0.20$) |
| $C_6$ | Retransmitir com probabilidade ($p = 0.10$) |
| $C_7$ | Situações em que os nós são bons retransmissores. No algoritmo *Epidemic* podem levar a alta sobrecarga de rede. Nunca retransmitir ($p = 0$) |

A logica de roteamento do algoritmo e *Epidemic* com classificação (*Epidemic+C*

pode se resumir no algoritmo (4.2):

**Algoritmo 4.2 Algoritmo *Epidemic+C* para retransmitir mensagens ao estabelecer uma conexão.**

**Entrada:** Nova conexão $x$, coleção de mensagens a retransmitir $M$, vetor de probabilidades $V$
**Saída:** Ação $a \in A = \{$ iniciar retransmissão, não retransmitir $\}$

```
1: n ← outroNó(x)
2: se éNóDestino(n) então
3:    retorna iniciar retransmissão
4: fim se
5: C_i ← solicitarAtributoAnotação(n)
6: se C_i é C_7 então
7:    retorna não retransmitir
8: fim se
9: p ← obterProbabilidade(C_i,V)
10: rnd ← gerarNroAleatório( )
11: se rnd é menor ou igual p então
12:    para cada mensagem m_i ∈ M = {m_1,m_2,···,m_k} faça
13:       se nóAceitaMensagem(n,m_i) então
14:          retorna iniciar retransmissão
15:       fim se
16:    fim para
17: senão
18:    retorna não retransmitir
19: fim se
```

O funcionamento deste algoritmo é mais simples. Este algoritmo usa um vetor de probabilidade de retransmissão, com valores fixos de probabilidade de retransmissão para cada classe de anotação (ver (4.1)). Dado um nó $x$, executando o algoritmo *Epidemic+C*, que estabeleça conexão com outro nó $y$, o funcionamento do algoritmo é o seguinte: se $x$ tiver uma mensagem com destino ao nó $y$, o nó $x$ decide iniciar a transmissão da mensagem (linhas 2 a 4). Posteriormente, nó $x$ solicita ao nó $y$ o atributo de anotação de classe dele (linha 5). Neste momento $y$ descreve a situação na qual transmitiria usando o conjunto de atributos e, consulta ao seu classificador treinado o atributo de anotação de classe. Quando o nó $x$ receber a classe de anotação do nó $y$, se o valor for $C_7$, significa que nessa zona e nesse momento o nó $y$ pode ser um bom disseminador de dados. Esta característica, embora seja desejável em outros algoritmos, no algoritmo *Epidemic* pode gerar situações de alta sobrecarga de rede. Contudo, decidiu-se excluir do processo de retransmissão, ao nós com maior capacidade de espalhamento de cópias de mensagens (valor de anotação de classe $C_7$, na linhas 6 a 8). Se o valor de anotação de classe for outro, o nó $x$ consulta ao

vetor de probabilidade $V$ o valor de probabilidade de entrega relacionado ao valor do atributo de anotação de classe e, atribui esse valor a $p$ (linha 9). Posteriormente, o nó $x$ tenta de enviar alguma das mensagens que possui ao nó $y$, com uma probabilidade de $p$. A retransmissão é iniciada quando o nó $y$ aceita alguma das mensagens de $x$ (linha 12 a 14). A mensagem poderia ser rejeitada por o nó $y$, se o nó já possuísse a mensagem.

## 4.3 Experimentos e resultados

Para a execução de todas as simulações, tanto para coleta de dados como para o teste do roteamento com classificação, utilizaram-se os parâmetros de simulação descritos na Tabela (3.1) do Capítulo (3) (pagina 55), do cenário com *traces* de movimento. A escolha dos nós origens e destinos de mensagens é realizada aleatoriamente de entre 13 nós, os quais permanecem dentro da área de estudo durante as cinco horas de simulação. Para o algoritmo *Epidemic*, realizou-se duas simulações, uma para coleta de dados e outra para o teste do algoritmo com classificação (*Epidemic+C*). Para o algoritmo *SaW*, uma vez que o algoritmo precisa da configuração do número inicial de cópias para replicar (chama-se de $L$), realizaram-se um total de sete simulações. A primeira simulação foi para a coleta de dados com um valor de $L = 300$. Usou-se um valor alto de $L$ porque a coleta de tuplas com um valor baixo de $L$ diminuiria rapidamente e, o dados coletados seriam de entregas de mensagens por *direct transmission* e não transmissões de replicação de cópias. As seis simulações restantes foram para o teste do algoritmo com classificação *SaW+C*, com valores de $L$ diferentes, $L = \{5, 10, 30, 50, 100, 150\}$.

Na Figura (4.6) mostram-se os resultados obtidos para o algoritmo *SaW* e a sua versão com classificação *SaW+C*.

Observa-se que para números de valores de cópia baixos, o aprimoramento é maior. Assim, para um valor de cópias de 5, obteve-se uma melhoria de 2% na probabilidade de entrega e uma diminuição na média do atraso de envio de 231 segundos. Aprecia-se uma leve melhoria no *overhead* de rede, porém, não significativa uma vez que o algoritmo original já apresenta baixa sobrecarga de rede (*overhead*), quando comparado com os outros algoritmos. A métrica de desempenho mais aprimorada foi a média do atraso, com valores de: 231.3, 25.2, 23.5, 24.2, 3.0 e 4.1 segundos, para os valores de cópia de: 5, 10, 30, 50, 100, 150 respectivamente.

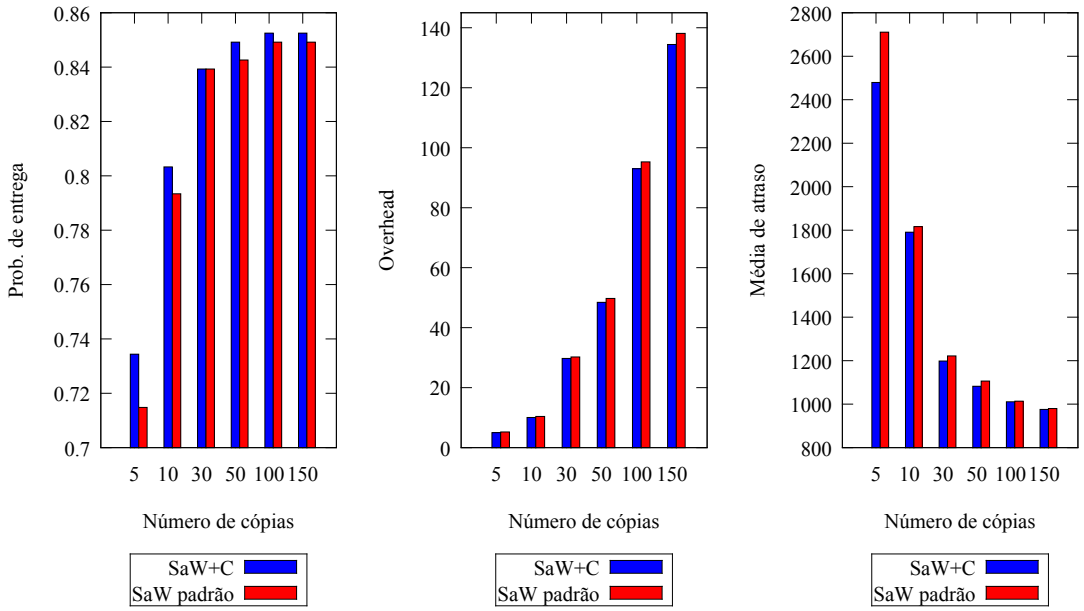



**Figura 4.6: Probabilidade de entrega, *overhead* e atraso do algoritmo *SaW* com e sem classificação para vários valores de número de cópias.**

Na Figura (4.7) mostram-se os resultados obtidos para o algoritmo *Epidemic* e a sua versão com classificação *Epidemic+C*.

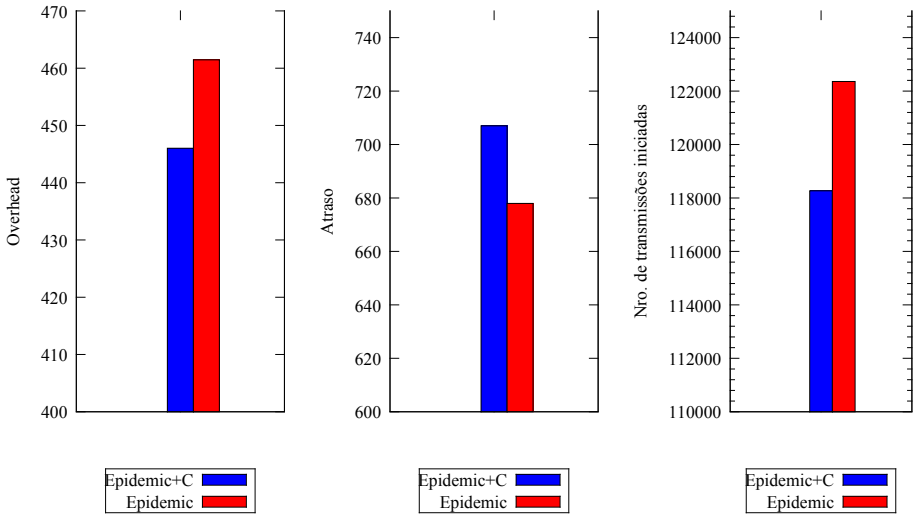



**Figura 4.7: *Overhead*, atraso e número de transmissões do algoritmo *Epidemic* com e sem classificação para vários valores de número de cópias.**

Observa-se uma diminuição dos valores de *overhead* e do número de transmissões utilizadas, em valores de 15.44 e 4089 respectivamente; não obstante, com um ganho de média de atraso de 29.04 segundos. A probabilidade de entrega manteve-se igual, com um valor de 0.8656, ou seja 86.56% de probabilidade de entrega.

## 4.4 Conslusões

Nesta seção foi apresentada uma abordagem de classificação de eventos de retransmissão de mensagens para aprimorar o desempenho dos algoritmos de roteamento por replicação de mensagens *SaW* e *Epidemic*. A abordagem proposta neste trabalho prediz a qualidade de um nó como retransmissor para os diferentes algoritmos avaliados.

A abordagem proposta foi aplicada a dois algoritmos de roteamento (*SaW* e *Epidemic*) utilizando *traces* de movimento real. Estes algoritmos agem diferentemente: enquanto o algoritmo *Epidemic* é efetivo em ambientes de redes esparsas ou de baixa densidade, o algoritmo *SaW* é efetivo em ambientes densos e de alta conectividade. A abordagem proposta adapta os algoritmos para não retransmitir em casos inconvenientes, segundo a lógica de cada algoritmo. Em ambos os casos, o roteamento realiza-se com menos quantidade de transmissões sem prejudicar a probabilidade de entrega.

Comparando o desempenho do algoritmo *SaW* e a sua versão com classificação *SaW+C*, o *SaW+C* apresentou um desempenho superior ou igual ao do *SaW*. A melhoria do desempenho foi mais significativa para os cenários onde o número de cópias foi baixo (5 ou 10 cópias).

Na avaliação do desempenho para o algoritmo *Epidemic*, o método proposto diminuiu o *overhead* da rede. No entanto, observou-se um aumento no atraso de envio de 29 segundos em média. Isto se deve ao fato de que o método proposto (*Epidemic+C*) foi projetado para excluir do processo de retransmissão a aqueles nós que retransmitem em menor tempo, uma vez que também geram maior *overhead* de rede.

# Capítulo 5

## Conclusões e linhas de pesquisa

Neste trabalho de mestrado foi projetado um método que utiliza o algoritmo de classificação baseado em árvores de decisão *C4.5*, para aprimorar o desempenho dos algoritmos de roteamento por replicação de mensagens. O método consiste em treinar um classificador com informação de eventos passados de retransmissões de mensagens e, posteriormente, decidir se repassar ou não as cópias das mensagens segundo a predição do classificador. A predição do classificador estima quão favorável pode ser a retransmissão de uma mensagem para um nó retransmissor, segundo determinadas condições. As situações em que um nó retransmite podem ser descritas usando atributos que representem a densidade de nós na localização do retransmissor, o intervalo de tempo em que retransmite e a zona geográfica onde o nó está localizado.

Do manifestado no Capítulo (2), pode se afirmar que:

- *VANETs* são redes de conectividade intermitente e topologia altamente dinâmica, nas quais, para a transferência de dados de aplicações tolerantes ao atraso, pode se utilizar o paradigma de roteamento baseado em armazenamento de mensagens (*store-carry-and-forward*), para incrementar a probabilidade de entrega, porém, incrementando também o atraso de envio.

- Para reduzir o atraso de envio em redes sem limitações críticas de recursos de armazenamento e energia, os algoritmos podem disseminar cópias das mensagens. A maior disseminação de cópias menor o atraso, não obstante, a sobrecarga da rede pela transferência das cópias é considerável em cenários de alta conectividade.

- Uma vez que cenários com redes altamente conectadas criam muita sobrecarga de rede, alguns algoritmos foram projetados para cenários específicos de conectividade

(redes esparsas ou densas). Porém, em cenários reais, como cidades, a distribuição dos nós no espaço não é uniforme e criam-se zonas de diferente densidade. Deste modo, tem-se a necessidade de adaptar o comportamento dos algoritmos a diferentes condições de densidade e conectividade para retransmitir cópias segundo seja necessário.

Do estudo de conectividade e avaliação do desempenho dos algoritmos em cenários de diferentes modelos de movimento (3), pode se concluir que:

- Modelos de movimento sintético, cujo movimento dos nós não apresenta ou apresenta poucas restrições, diminuem o impacto da sobrecarga de rede dos algoritmos baseados em replicação de mensagens. Isto porque em cenários reais, como o movimento veicular em cidades, a limitação do movimento em algumas zonas criam componentes de redes de grande tamanho e conectividade. Por exemplo, na Seção (3.4.3) o cenário dos *traces* de movimento revela excessiva sobrecarga para o protocolo *Epidemic*, aproximadamente ao dobro quando comparado com os resultados nos cenários dos outros modelos de movimento.

- Na avaliação da sobrecarga de rede dos algoritmos de roteamento baseados em replicação de mensagens, os modelos de movimento sintético sim podem determinar qual algoritmo é melhor do que outro, porém, são imprecisos em quantificar a diferença entre um e outro algoritmo. Por exemplo, nos dois cenários de movimento sintético, o algoritmo *SaW* gera uma sobrecarga menor em 100 aproximadamente do que no algoritmo *Epidemic*. Esta diferencia poderia se considerar não significativa dado que o atraso usando *SaW* é maior do que usando *Epidemic*. Não obstante, o cenário dos *traces* reais revela uma diferença em sobrecarga de mais de 250.

Dos resultados obtidos pela aplicação do método proposto no cenário dos *traces* de movimento real pode se afirmar que:

- Segundo as tuplas de atributos coletadas para o treinamento do classificador, com relação aos valores de *lobby index* conclui-se que:
  - Permite distinguir zonas de diferente densidade de nós, e assim, de maior congestão veicular.
  - Está relacionado com o número total de nós na rede, que é variável e vá mudando com o tempo.

  - Está relacionado com o tempo que demora a retransmissão de uma mensagem recebida. Por exemplo, nós com *lobby index* alto retransmitem em menos tempo do nós com *lobby index* baixo.
  - Está relacionado com a distância à que os nós retransmitem uma mensagem recebida. Por exemplo, nós com *lobby index* alto retransmitem a uma distância menor do que nós com *lobby index* baixo.

- O método é capaz de predizer situações de retransmissão não favoráveis para os algoritmos avaliados: *SaW* e *Epidemic*. Desta maneira, os nós realizam menos transmissões de cópias, não retransmitindo quando não é favorável, sem diminuir a probabilidade de entrega.

- No caso de algoritmo *SaW+C*, a redução da sobrecarga não é muito significante porque este algoritmo já possui um número fixo de contagem de réplicas de uma mensagem. Não obstante, uma vez que a escolha dos retransmissores é seletiva, os nós podem aproveitar melhor o número de contagem de cópias configurado, e assim, alguns nós podem encontrar ao destino sem precisar realizar *direct transmission*. A pequena diferença entre a sobrecarga de *Saw* e *SaW+C*, atribui-se a este evento.

- No caso do algoritmo *Epidemic*, denegam-se as retransmissões a nós em situações de alta conectividade, que retransmitem rapidamente e contribuem à inundação de mensagens. Assim, conseguiu-se diminuir a sobrecarga em 15.4 e o número de transmissões em 4000, sem diminuir a probabilidade de entrega, porém, incrementando o a média do atraso em 29 segundos.

### 5.0.1 Trabalhos futuros

Os algoritmos de classificação podem ser utilizados no projeto de uma variedade de algoritmos de roteamento, uma vez que, podem servir para predizer diferentes tipos de eventos, os quais podem ser descritos ou relacionados usando diferentes atributos de classificação. Adicionalmente, consideram-se as seguintes possíveis aplicações:

- Qualidade de nós em redes móveis heterogêneas: Algoritmos de classificação podem ser utilizados para a classificação de nós em redes heterogêneas. Neste tipo de redes a qualificação dos nós segundo as suas características de movimento e recursos pode ser aproveita para diminuir a carga em nós de baixa capacidade, em um esquema cooperativo de transmissão de mensagens.

- *Tuning* de valores de configuração de algoritmos de roteamento. Um exemplo desta aplicação seria que utilizando informação de retransmissões passadas e, utilizando tuplas de atributos que descrevam diferentes situações, poder-se-ia estimar o tempo que precisa uma mensagem para chegar ao destino. Assim, poder-se-ia aplicar medidas de controle de $TTL$ para diminuir a redundância de copias da mensagens sem prejudicar o desempenho, e com valores de $TTL$ que estime o classificador.

# Referências


AHMED, S.; KANHERE, S. A bayesian routing framework for delay tolerant networks. In: *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*. [S.l.: s.n.], 2010. p. 1–6. ISSN 1525-3511.

BAI, F.; SADAGOPAN, N.; HELMY, A. Important: a framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks. In: *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*. [S.l.: s.n.], 2003. v. 2, p. 825–835 vol.2. ISSN 0743-166X.

BASAGNI, S.; CONTI, M.; GIORDANO, S. *Mobile Ad Hoc Networking*. [S.l.]: Wiley-IEEE Press, 2004. ISBN 0471373133.

BERNSEN, J.; MANIVANNAN, D. Unicast routing protocols for vehicular ad hoc networks: A critical comparison and classification. *Pervasive and Mobile Computing*, v. 5, n. 1, p. 1 – 18, 2009. ISSN 1574-1192.

BRANDES, U. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, v. 25, p. 163–177, 2001.

BURGESS, J. et al. Maxprop: Routing for vehicle-based disruption-tolerant networks. In: *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*. [S.l.: s.n.], 2006. p. 1–11. ISSN 0743-166X.

CASTEIGTS, A.; NAYAK, A.; STOJMENOVIC, I. Communication protocols for vehicular ad hoc networks. *Wirel. Commun. Mob. Comput.*, John Wiley and Sons Ltd., Chichester, UK, v. 11, n. 5, p. 567–582, maio 2011. ISSN 1530-8669.

CHENG, P. C. et al. GeoDTN+Nav: Geographic DTN Routing with Navigator Prediction for Urban Vehicular Environments. *Mob. Netw. Appl.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 15, p. 61–82, fev. 2010. ISSN 1383-469X. Disponível em: <http://dx.doi.org/10.1007/s11036-009-0181-6>.

COLAGROSSO, M. A classification approach to broadcasting in mobile ad hoc network. In: *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*. [S.l.: s.n.], 2005. v. 2, p. 1112–1117 Vol. 2.

COSTA, L. et al. Characterization of complex networks: A survey of measurements. *Advances in Physics*, Taylor & Francis, v. 56, p. 167–242, 2006.

DIKAIAKOS, M. D. et al. Vitp: an information transfer protocol for vehicular computing. In: *in: VANET 05: Proceedings of the 2nd ACM International Workshop on Vehicular Ad Hoc Networks, ACM*. [S.l.]: Press, 2005.

DOERING, M. et al. A new mobility trace for realistic large-scale simulation of bus-based dtns. In: *Proceedings of the 5th ACM workshop on Challenged networks*. New York, NY, USA: ACM, 2010. (CHANTS '10), p. 71–74. ISBN 978-1-4503-0139-8.

EENENNAAM, E. M. V. et al. Exploring the solution space of beaconing in vanets. In: *First IEEE Vehicular Networking Conference, VNC2009, Tokyo, Japan*. [S.l.]: IEEE Communications Society, 2009. p. 26:1–26:8.

EKMAN, F. et al. Working day movement model. In: . [S.l.: s.n.], 2008. p. 33–40. Cited By (since 1996) 1.

FRANCK, L.; GIL-CASTINEIRA, F. Using delay tolerant networks for car2car communications. In: *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*. [S.l.: s.n.], 2007. p. 2573 –2578.

GAO, S.; ZHANG, L.; ZHANG, H. Energy-aware spray and wait routing in mobile opportunistic sensor networks. In: . [S.l.: s.n.], 2010. p. 1058–1063. Cited By (since 1996) 0.

GUNASEKARAN, R.; MAHENDRAN, V.; MURTHY, C. Performance modeling of delay tolerant network routing via queueing petri nets. In: . [S.l.: s.n.], 2012. Cited By (since 1996) 0.

HAAS, Z.; SMALL, T. A new networking model for biological applications of ad hoc sensor networks. *IEEE/ACM Trans. Netw.*, IEEE Press, Piscataway, NJ, USA, v. 14, n. 1, p. 27–40, fev. 2006. ISSN 1063-6692.

HALL, M. et al. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, ACM, New York, NY, USA, v. 11, n. 1, p. 10–18, nov. 2009. ISSN 1931-0145.

HARRAS, K.; ALMEROTH, K.; BELDING-ROYER, E. Delay tolerant mobile networks (dtmns): controlled flooding in sparse mobile networks. In: *Proceedings of the 4th IFIP-TC6 international conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communication Systems*. Berlin, Heidelberg: Springer-Verlag, 2005. (NETWORKING'05), p. 1180–1192. ISBN 3-540-25809-4, 978-3-540-25809-4.

HARTENSTEIN, H.; LABERTEAUX, K. A tutorial survey on vehicular ad hoc networks. *Communications Magazine, IEEE*, v. 46, n. 6, p. 164 –171, june 2008. ISSN 0163-6804.

HARTENSTEIN, H.; LABERTEAUX, K.; EBRARY, I. *VANET: vehicular applications and inter-networking technologies*. Wiley Online Library, 2010. Disponível em: <http://onlinelibrary.wiley.com/doi/10.1002/9780470740637.fmatter/summary>.

HULL, B. et al. Cartel: a distributed mobile sensor computing system. In: *Proceedings of the 4th international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2006. (SenSys '06), p. 125–138. ISBN 1-59593-343-3. Disponível em: <http://doi.acm.org/10.1145/1182807.1182821>.

JETCHEVA, J. et al. Design and evaluation of a metropolitan area multitier wireless ad hoc network architecture. In: . [S.l.: s.n.], 2003. p. 32–43.

KARAGIANNIS, G. et al. Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions. *Communications Surveys Tutorials, IEEE*, v. 13, n. 4, p. 584 –616, quarter 2011. ISSN 1553-877X.

KARP, B.; KUNG, H. T. Gpsr: greedy perimeter stateless routing for wireless networks. In: *Proceedings of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2000. (MobiCom '00), p. 243–254. ISBN 1-58113-197-6.

KERäNEN, A.; KäRKKäINEN, T.; OTT, J. Simulating mobility and dtns with the one. *Journal of Communications*, v. 5, n. 2, p. 92–105, 2010. Cited By (since 1996) 10.

KERäNEN, A.; OTT, J.; KäRKKäINEN, T. The one simulator for dtn protocol evaluation. In: *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009. (Simutools '09), p. 55:1–55:10. ISBN 978-963-9799-45-5.

KORN, A.; SCHUBERT, A.; TELCS, A. Lobby index in networks. *Physica A: Statistical Mechanics and its Applications*, v. 388, n. 11, p. 2221 – 2226, 2009. ISSN 0378-4371.

LEE, U. et al. Code torrent: content distribution using network coding in vanet. In: *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*. New York, NY, USA: ACM, 2006. (MobiShare '06), p. 1–5. ISBN 1-59593-558-4.

LI, M.; YANG, Z.; LOU, W. Codeon: Cooperative popular content distribution for vehicular networks using symbol level network coding. *Selected Areas in Communications, IEEE Journal on*, v. 29, n. 1, p. 223 –235, january 2011. ISSN 0733-8716.

LIM, T.-S.; LOH, W.-Y.; SHIH, Y.-S. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Mach. Learn.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 40, n. 3, p. 203–228, set. 2000. ISSN 0885-6125. Disponível em: <http://dx.doi.org/10.1023/A:1007608224229>.

LINDGREN, A.; DORIA, A.; SCHELÉN, O. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 7, n. 3, p. 19–20, jul. 2003. ISSN 1559-1662.

LOCHERT, C. et al. A routing strategy for vehicular ad hoc networks in city environments. In: *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*. [S.l.: s.n.], 2003. p. 156–161.

MITROKOTSA, A.; DIMITRAKAKIS, C. Intrusion detection in manet using classification algorithms: The effects of cost and model selection. *Ad Hoc Netw.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 11, n. 1, p. 226–237, jan. 2013. ISSN 1570-8705. Disponível em: <http://dx.doi.org/10.1016/j.adhoc.2012.05.006>.

MOGHADAM, A.; SCHULZRINNE, H. Interest-aware content distribution protocol for mobile disruption-tolerant networks. In: . [S.l.: s.n.], 2009. Cited By (since 1996) 0.

NELSON, S.; BAKHT, M.; KRAVETS, R. Encounter-based routing in dtns. In: *INFOCOM 2009, IEEE*. [S.l.: s.n.], 2009. p. 846–854. ISSN 0743-166X.

NEWMAN, M. J. A measure of betweenness centrality based on random walks. *Social Networks*, v. 27, n. 1, p. 39 – 54, 2005. ISSN 0378-8733.

PALLIS, G. et al. On the structure and evolution of vehicular networks. In: *Modeling, Analysis Simulation of Computer and Telecommunication Systems, 2009. MASCOTS '09. IEEE International Symposium on*. [S.l.: s.n.], 2009. p. 1–10. ISSN 1526-7539.

QUINLAN, J. R. *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN 1-55860-238-0.

SOMMER, C.; TONGUZ, O.; DRESSLER, F. Adaptive beaconing for delay-sensitive and congestion-aware traffic information systems. In: *Vehicular Networking Conference (VNC), 2010 IEEE*. [S.l.: s.n.], 2010. p. 1–8. ISSN 2157-9857.

SPYROPOULOS, T.; PSOUNIS, K.; RAGHAVENDRA, C. S. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In: *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. New York, NY, USA: ACM, 2005. (WDTN '05), p. 252–259. ISBN 1-59593-026-4.

SPYROPOULOS, T. et al. Routing for disruption tolerant networks: taxonomy and design. *Wirel. Netw.*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 16, n. 8, p. 2349–2370, nov. 2010. ISSN 1022-0038.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN 0321321367.

TONGUZ, O.; WISITPONGPHAN, N.; BAI, F. Dv-cast: A distributed vehicular broadcast protocol for vehicular ad hoc networks. *Wireless Communications, IEEE*, v. 17, n. 2, p. 47–57, 2010. ISSN 1536-1284.

*Broadcasting in VANET*. 7–12 p. Disponível em: <http://dx.doi.org/10.1109/move.2007.4300825>.

VAHDAT, A.; BECKER, D. *Epidemic routing for partially-connected ad hoc networks*. [S.l.], 2000.

VASILAKOS, A.; ZHANG, Y.; SPYROPOULOS, T. *Delay Tolerant Networks: Protocols and Applications*. Hoboken: CRC Press, 2011. (Wireless Networks and Mobile Communications).

VENDRAMIN, A. et al. Grant: Inferring best forwarders from complex networks' dynamics through a greedy ant colony optimization. *Computer Networks*, v. 56, n. 3, p. 997–1015, 2012. Cited By (since 1996) 2.

WEHRLE, K.; GROSS, J.; GÜNES, M. *Modeling and Tools for Network Simulation*. [S.l.]: Springer-Verlag Berlin Heidelberg, 2010. ISBN 9783642123313 3642123317.

WISCHHOF, L.; EBNER, A.; ROHLING, H. Information dissemination in self-organizing intervehicle networks. *Intelligent Transportation Systems, IEEE Transactions on*, v. 6, n. 1, p. 90–101, 2005. ISSN 1524-9050.

WOLTERINK, W.; HEIJENK, G.; KARAGIANNIS, G. Information dissemination in vanets by piggybacking on beacons - an analysis of the impact of network parameters. In: *Vehicular Networking Conference (VNC), 2011 IEEE*. [S.l.: s.n.], 2011. p. 94–101. ISSN 2157-9857.

YIN, L.; LU, H.-M.; CAO, Y.-D. A novel single copy replication routing strategy for delay tolerant networks. In: . [S.l.: s.n.], 2009. Cited By (since 1996) 0.

YU, W.; WU, C.; HU, X. Spray and routing for message delivery in challenged networks. In: . [S.l.: s.n.], 2010. p. 472–475. Cited By (since 1996).

ZEADALLY, S. et al. Vehicular ad hoc networks (vanets): Status, results, and challenges. *Telecommunication Systems*, v. 50, n. 4, p. 217–241, 2012.

ZHANG, X. et al. Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing. In: *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2007. (MobiCom '07), p. 195–206. ISBN 978-1-59593-681-3.

ZHANG, Y.; ZHAO, J.; CAO, G. Roadcast: A popularity aware content sharing scheme in vanets. In: *Distributed Computing Systems, 2009. ICDCS '09. 29th IEEE International Conference on*. [S.l.: s.n.], 2009. p. 223 –230. ISSN 1063-6927.

# Glossário

**API** – *Application Programming Interface*

**DTNs** – *Delay and Disruption Tolerant Networks*

**GUI** – *Graphical User Interface*

**MANETs** – *Mobile ad hoc networks*

**Prophet** – *Probabilistic ROuting Protocol using History of Encounters and Transitivity*

**RWP** – *Modelo de movimento Random Waypoint*

**SPMB** – *Modelo de movimento Shortest Path Map Based Movement*

**SaW** – *Algoritmo de roteamento Spray and Wait*

**TTL** – *Time To Live*

**The ONE** – *The Opportunistic Networking Environment Simulator*

**V2I ou I2V** – *Comunicações de veículo a infraestrutura ou de infraestrutura a veículo*

**V2V** – *Comunicações veículo a veículo*

**VANETs** – *Vehicular Ad hoc Networks*

**Weka** – *Waikato Environment for Knowledge Analysis*

# Anexo A

# Descrição do processo de simulação dirigido ao usuário desenvolvedor

## A.1 Descrição do ciclo de simulação

A classe *DTNSIM* é a classe principal e a que inicia o processo de simulação. Para explicar o funcionamento do simulador divide-se o ciclo de simulação em três fases: (i) fase de carga de dados e configuração, (ii) fase de atualização e, (iii) fase de finalização. Em cada uma destas fases incluem-se só as classes mais importantes, isto é, aquelas classes que o usuário desenvolvedor terá que revisar ou modificar para a programação de novas funcionalidades no simulador.

**Fase de carga de dados e configuração:**Nesta fase realizam-se as tarefas de leitura de arquivos de cofiguração, leitura de arquivos de dados, criação de objetos de simulação e agendamento da ocorrência de eventos. O diagrama de sequencia da Figura (A.1) corresponde a um exemplo de execução do simulador utilizando execução por indexação ou *run indexing*.

No primeiro passo da Figura (A.1), o simulador lê os arquivos de configuração para obter os valores das propriedades ou atributos de simulação e conservá-los em memória em um objeto da classe *Settings*. Quando se utiliza *run indexing* e se configura vários valores para uma propriedade, a classe *Settings* representa essas configurações como cadeias de valores separados por o caráter ponto e vírgula. Para obter os valores de configuração relacionados a cada cenário segundo o índice de execução, o simulador entra em um *loop* de execução.No *loop* de execução, o método *setRunIndex* retorna o número de simulação que se está realizando e, a classe *Settings*,

realiza um *parsing* das cadeias de configuração para obter os valores de configuração segundo o índice de execução indicado. Desta forma, no momento de entrar ao *loop* de execução, obtém-se as configurações para cada cenário e executa-se a simulação respectiva segundo o indicado pelo índice de execução.

Já dentro do *loop de execução*, no passo dois da Figura (A.1), inicia-se o modelo de simulação para a execução atual e criam-se e configuram-se os objetos de simulação com os atributos obtidos pela classe *Settings*. O primeiro objeto em criar-se é o objeto *SimScenario* que representa o caso de estudo ou cenário de simulação. Este objeto é inicializado com vários valores, entre os quais, está o intervalo de atualização (*updateInterval*). Posteriormente o objeto da classe *SimScenario* cria e configura outros objetos como:

- •Gerenciador de eventos externos: Agenda a sequência de eventos a executar em cada intervalo de atualização, na fase de atualização.
- •Nós ou *host* da rede: Configuram-se segundo os valores para as propriedades de cada grupo de *hosts* no arquivo de configuração. No simulador, os nós estão representados pela classe *DTNHost*. Porém, no arquivo de configuração o *namespace Group* agrupa configurações de rede e movimento para todos os nós que pertencem ao grupo. Embora o *namespace Group* não tenha representação como classe no código fonte do simulador, serve para facilitar a criação e atribuição de diferentes configurações para os nós de um mesmo grupo. Inicialmente criam-se objetos com configurações padrão para cada nó que pertence ao grupo, por exemplo: interfaces de rede, objeto de modelo de movimento, objeto de aplicação. Depois, para cada nó que pertence a um determinado grupo, replicam-se as configurações dos objetos padrão do grupo.
- •Objeto mundo da classe *World*: Inclui as propriedades do lugar físico onde atua a *DTN*. É um objeto muito importante porque a ele se atribuem vários outros objetos relacionados à simulação de todo o ambiente.

Esta fase termina com a criação de todos os objetos de relatórios que se indicaram no arquivo de configuração. Posteriormente, obtém-se uma referencia ao mundo ou ambiente de simulação e inicia a fase de atualização.

**Fase de atualização:**Nesta fase o simulador executa progressivamente a simulação (em cada intervalo de atualização), atualizando diferentes objetos e realizando eventos programados na fase de configuração. A classe de interfaz de usuário (*DTNSimTextUI* de entorno de consola ou *DTNSimGUI* de entorno GUI) inicia o a reprodução

de uma simulação com o método *runSim* e, a partir deste momento, chama-se ao método *update* em cada intervalo de tempo de atualização até finalizar o tempo de simulação. A partir de este momento se inicia uma cadeia de execução dos métodos *update* das classes dos objetos com comportamento dinâmico, como se mostra na Figura (A.2).

O objeto de classe *World* representa o mundo ou ambiente físico onde tudo acontece. A atualização do seu estado (execução do seu método *update*), atualiza o estado de todos os objetos simulados da seguinte maneira: primeiro se verifica se algum evento externo está programado para simulação; se tiver algum, o executa e atualiza o estado de todos os nós da rede porque eles podem ter sido afetados pela ocorrência desse evento. Posteriormente agenda futuros eventos externos e atualiza a novas localizações de todos os nós com o método *moveHosts*. Quando os nós se movimentam podem acontecer novos eventos (modificação dos estados de conectividade, mensagens abortados, etc); por este motivo o objeto World realiza uma vez mais uma nova atualização do estado dos nós da rede.

A atualização do estado dos nós modifica o estado dos elementos que lhe pertencem, como o roteador, a interfaz de rede ou a aplicação que o nó executa (Figura (A.3)). A classe *MessageRouter* é a classe que representa o módulo que encaminha as mensagens e funciones básicas de roteamento, duas classes entendem dela: (i) a classe *AtiveRouter*, para simular algoritmos de roteamento no simulador; e (ii) a classe *PassiveRouter*, usado para interagir com simuladores externos de roteamento para *DTNs*. Todo algoritmo de roteamento *DTN* projetado para o simulador deve estender a classe *AtiveRouter*. O funcionamento de cada algoritmo é determinado no seu método *updade*, o qual terá que ser sobre-escrito em cada algoritmo que estenda a classe*AtiveRouter*. Adicionalmente, pode se sobre-escrever outros métodos para coletar informação de roteamento, ou executar cálculos com o fim de aprimorar o desempenho.

**Fase de finalização:**Por último, quando a classe de interfaz de usuário (*DTNSimTextUI* ou *DTNSimGUI*) determinar que o tempo de simulação acabou, os dados e métricas coletados para relatórios serão salvados em arquivos para análises e visualização.

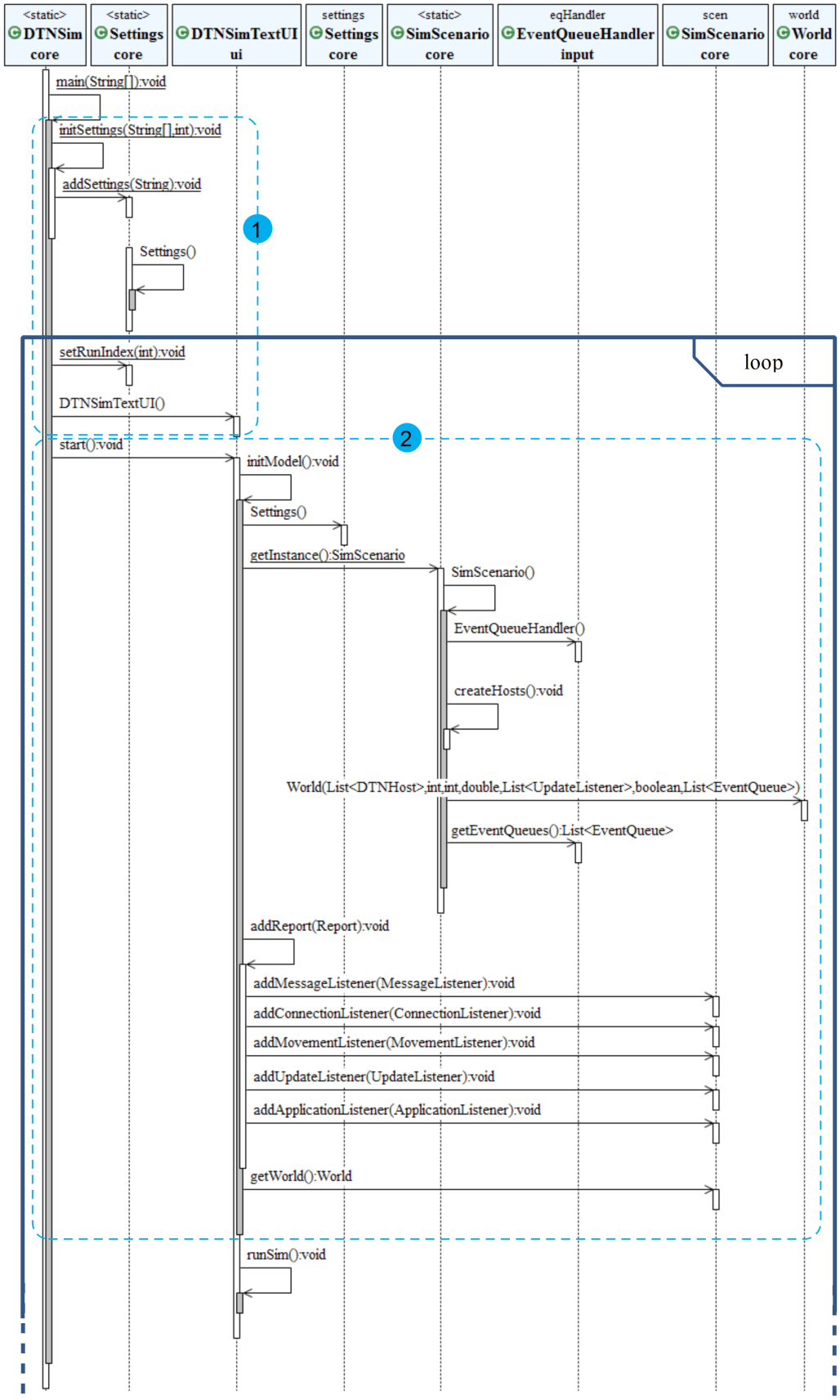



**Figura A.1: Diagrama de sequencia de configuração previa à simulação.**

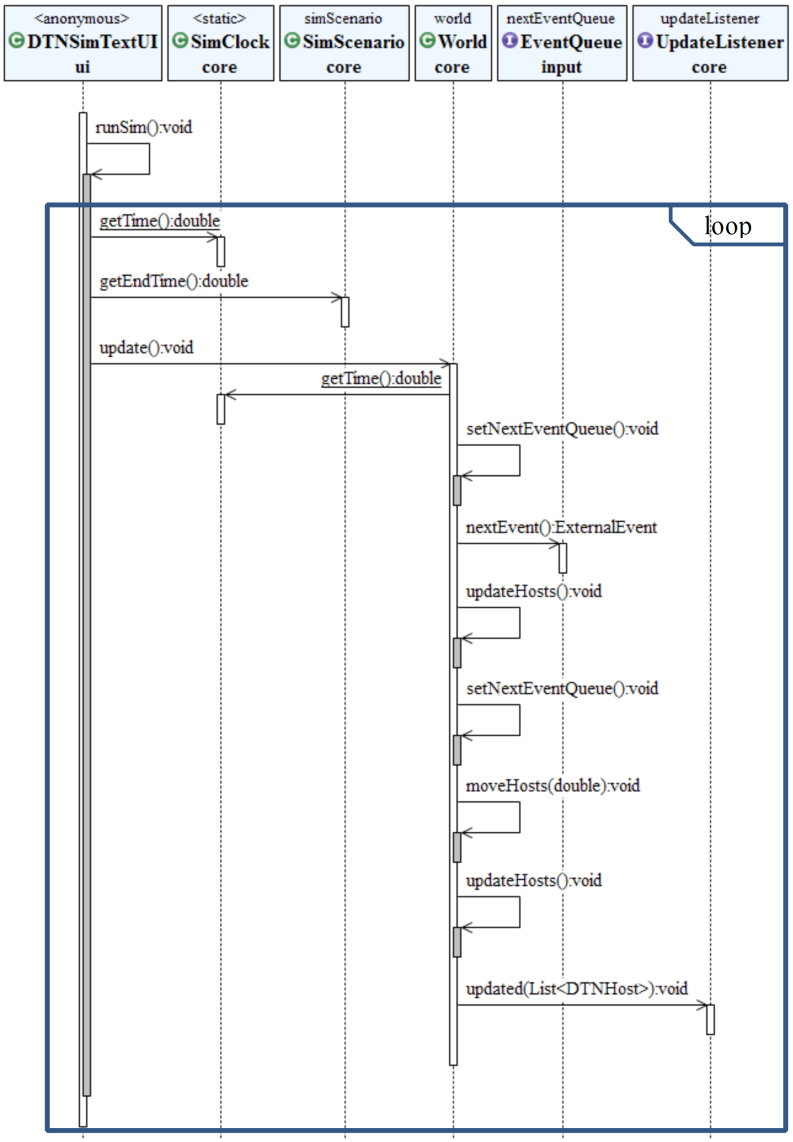



**Figura A.2: Digrama de sequencia da atualização do cenário simulado.**

## A.2 Ativação e desativação dos radiotransmissores dos nós

A funcionalidade para modificar a atividade de rádio é fornecida no simulador, porém, para todos os nós de um grupo. Isto significa que indicando uma sequência de faixas de tempo de atividade no arquivo de configuração, pode-se modificar a atividade de rádio de todo um grupo de nós. Não obstante, para programar esta funcionalidade para nós individuais, é necessário modificar o simulador. A modificação da atividade do radio-transmissor para grupos de nós é realizada através do objeto da classe *ActivenessHandler*, que está incluído como atributo no objeto de interface de rede *NetworkInterface*. Este objeto de interface de rede, é um objeto padrão criado com as configurações básicas, para

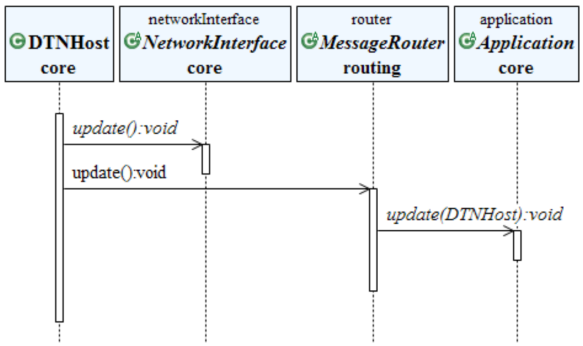



**Figura A.3: Digrama de sequencia da atualização do estado de nós.**

que posteriormente, cada nó que pertença ao grupo, possa replicar as configurações desse objeto no próprio objeto de interface de rede. Para permitir que cada nó possua os seus próprios intervalos de tempo de atividade de rádio, é necessário que no momento da replicação das configurações por padrão para cada grupo, não se replique os intervalos de tempo de atividade de rádio, senão, cria-se um objeto *ActivenessHandler* com configurações próprias. Assim, no momento de replicação, são enviados os tempos de atividade de rádio específicos para um nó e, em vez de replicar o objeto da classe *ActivenessHandler* que pertence ao objeto de interface de rede, cria-se um objeto novo que inclua intervalos de atividade específicos para esse nó. Para isto basta com criar um novo construtor para o objeto *ActivenessHandler* que aceite uma cadeia de intervalos de tempo de atividade, o procedimento restante é o mesmo que o realizado para a configuração por grupo, só que agora os intervalos enviados, são diferentes para cada nó.

## A.3 Criação de um repórter

A criação de um repórter é uma tarefa simples, porém, é necessário conhecer as interfaces de eventos e o procedimento necessário. Durante todo o processo de simulação acontecem diferentes eventos como: criação de uma mensagem, estabelecimento de uma conexão de rede, deslocamento de um nó. Os métodos que produzem estes eventos, informam do seu acontecimento a todos os objetos das classes que implementem determinadas interfaces. Cada uma de estas interfaces está relacionada a um tipo de evento, assim temos:

- ApplicationListener: Classes de relatórios que desejem receber eventos de aplicação deveriam implementar esta interface.

- ConnectionListener: Classes de relatórios de eventos de conectividade deveriam im-

plementar esta interface. Os de eventos de conectividade são: conexão e desconexão.

- MessageListener: Classes de relatórios de eventos de transmissões de mensagem deveriam implementar esta interface. Exemplos de eventos de transmissão são: transmissão iniciada, abortada, mensagem transmitido, criação de mensagem, entrega de mensagem.

- MovementListener: Classes de relatórios de eventos de movimento deveriam implementar esta interface. Os eventos de movimento são: novo localização e localização inicial.

- UpdateListener: Classes de relatórios de eventos de atualização deveriam implementar esta interface. Objetos que implementem esta classe podem atualizar diferentes dados e realizar cálculos em cada intervalo de atualização do simulador.

Contudo, cada classe de relatório pode implementar uma o mais de uma interfaz, usar os métodos indicados para o cálculo e o cadastro de métricas e, finalmente, quando acabar a simulação, o simulador produz um arquivo por cada relatório indicado no arquivo de configuração.