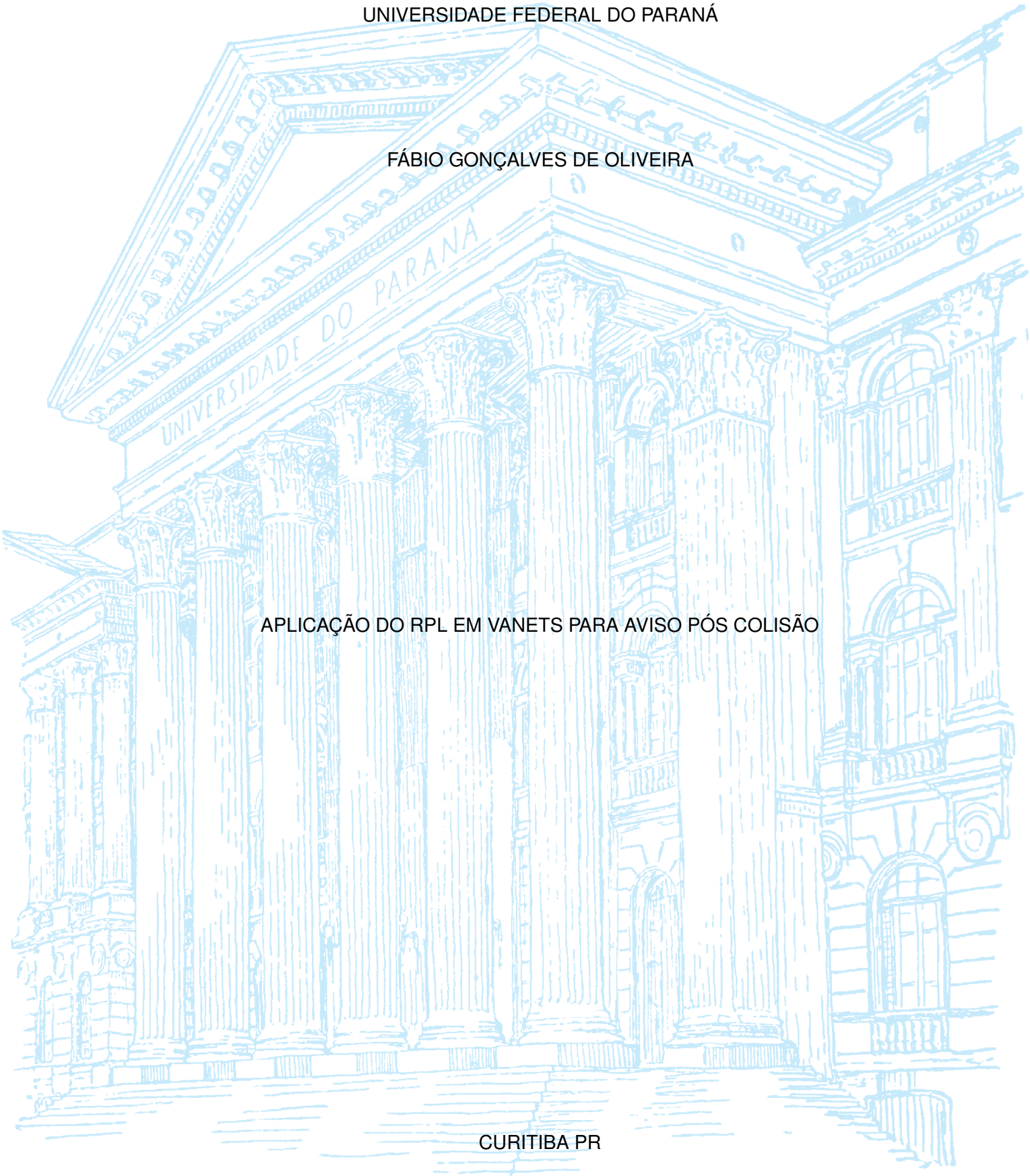
UNIVERSIDADE FEDERAL DO PARANÁ

FÁBIO GONÇALVES DE OLIVEIRA



APLICAÇÃO DO RPL EM VANETS PARA AVISO PÓS COLISÃO

CURITIBA PR

2024

FÁBIO GONÇALVES DE OLIVEIRA

APLICAÇÃO DO RPL EM VANETS PARA AVISO PÓS COLISÃO

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Engenharia Elétrica, Setor de Tecnologia, da Universidade Federal do Paraná.

Área de concentração: *Engenharia Elétrica*.

Orientador: Carlos Marcelo Pedroso.

CURITIBA PR

2024

AGRADECIMENTOS

Agradeço à minha família, principalmente aos meus pais, Zélia e Rogério, e aos meus irmãos, Felipe e Vinícius, por compreenderem a minha ausência e sempre me mostrarem a importância da educação. Agradeço à Lyvia, minha esposa, pelo apoio e paciência ao longo desta fase de maior exigência acadêmica e pela companhia ao longo das exaustivas noites de trabalho.

Agradeço aos amigos por deixarem essa caminhada mais leve e pelo apoio ao longo dos anos de faculdade. Agradeço ao professor orientador, Carlos Marcelo Pedroso, pelos rápidos retornos e por me auxiliar ao longo do desenvolvimento deste trabalho. Por fim, agradeço a todos que direta ou indiretamente colaboraram com este trabalho.

LISTA DE ACRÔNIMOS

ARSSI	*Average Received Signal Strength*
CH	*Cluster Head*
CM	*Cluster member*
DAO	*Destination Advertisement Object*
DAO-ACK	DAO *Acknowledgement*
DIO	DODAG *Information Object*
DIS	DODAG *Information Solicitation*
DODAG	*Destination Oriented Directed Acyclic Graph*
DRSC	*Dedicated Short-Range Communications*
eNB	*Evolved Node B*
ETX	*Expected Transmission Count*
IETF	*Internet Engineering Task Force*
IOV	*Internet of vehicles)*
IPV6	*Internet Protocol Version* 6
ITS	*Intelligent Transportation System*
LLN	*Low-Power and Lossy Network*
LoRaWan	*Long Range Wide Area Network*
LTE	*Long Term Evolution*
MANET	*Mobile Ad hoc Network*
MARPL	*Mobility Aware* RPL
MRHOF	*Minimum Rank Hysteresis Objective Function*
OBU	*On-Board Unit*
OF0	*Objective Function 0*
OMS	Organização Munidal da Saúde
PDR	*Packet Delivery Ratio*
POF	*Programable Objective Function*
RFC	*Request for Comments*
RPL	IPv6 *Routing Protocol for Low-Power and Lossy Networks*
RSSI	*Received Signal Strength Indication*
RSU	*Road-Side Unit*
SDN	*Software Defined Network*
TA	*Trusted Authority*
UDGM	*Unit Disk Graph Medium*
V2I	*Vehicle to Infrastructure*
V2P	*Vehicle to Pedestrian*

V2V	*Vehicle to Vehicle*
V2X	*Vehicle to Everything*
VANET	*Vehicular Ad Hoc Network*
WiMax	*Worldwide Interoperability for Microwave Access*

LISTA DE FIGURAS

LISTA DE SÍMBOLOS

c	Contador de inconsistências do algoritmo *trickle*
d	Distância entre nós
d_{max}	Distância máxima de transmissão do rádio
k	Constante de redundância do algoritmo *trickle*
I	Intervalo de tempo do algoritmo *trickle*
I_{max}	Intervalo máximo de tempo do algoritmo *trickle*
I_{min}	Intervalo mínimo de tempo do algoritmo *trickle*
P_{Rx}	Taxa de sucesso de recepção do modelo UDGM
t	Valor de tempo

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Estima-se que, em 2015, havia cerca de 1,1 bilhão de veículos de pequeno porte no mundo. Espera-se que até 2050 esse número atinja até 1,75 bilhão [1]. Esse aumento significativo na quantidade de veículos traz consigo uma preocupação relacionada à segurança. Conforme indicado no Relatório Global de Segurança Rodoviária publicada pela OMS (Organização Mundial da Saúde) em 2018, aproximadamente 1,35 milhão de vidas são perdidas anualmente em acidentes de trânsito, e até 50 milhões de pessoas sofrem ferimentos [2].

Neste contexto, tem se consolidado cada vez mais o conceito de ITS (*Intelligent Transportation System*), que utiliza uma variedade de tecnologias avançadas para melhorar a eficiência e, principalmente, a segurança das estradas [3]. Dentro desse cenário, as VANETs (*Vehicular Ad Hoc Networks*) têm emergido como uma abordagem promissora para aprimorar a segurança viária, permitindo a comunicação entre veículos e infraestruturas rodoviárias [4].

Estas redes têm o potencial de coletar e compartilhar informações críticas em tempo real, como condições de tráfego, eventos de emergência e alertas de segurança, contribuindo para a prevenção de acidentes e o aumento da eficácia das operações de gestão de tráfego [5]. No entanto, a eficácia das VANETs depende significativamente da eficiência dos protocolos de roteamento utilizados para encaminhar dados entre os nós da rede.

O presente trabalho tem como objetivo avaliar o desempenho do protocolo de roteamento RPL (IPv6 *Routing Protocol for Low-Power and Lossy Networks*) em cenários de notificação pós colisão veicular. Serão conduzidas simulações de cenários de colisão por meio do emulador Cooja, com o uso de firmwares desenvolvidos utilizando o sistema operacional Contiki.

Esta monografia está organizada da seguinte maneira: inicialmente, no capítulo 2, são apresentados os objetivos gerais e específicos da pesquisa e também a metodologia utilizada. Nesse contexto, são discutidos os recursos necessários para a realização das simulações, os tipos de testes realizados e como foi feita a análise do desempenho. Em seguida, no capítulo 3, é conduzida uma revisão de conceitos fundamentais, que engloba uma análise das Redes Veiculares *Ad Hoc*, do protocolo RPL, do sistema operacional Contiki e dos trabalhos mais recentes que se dedicam à notificação de colisões entre veículos. Posteriormente, no capítulo 4, são detalhados a aplicação desenvolvida, o modelo de mobilidade utilizado, os parâmetros de cada cenário, e são apresentados e discutidos os resultados obtidos. Por fim, no capítulo 5, são realizadas as conclusões do trabalho.

2 OBJETIVOS E METODOLOGIA

Neste capitulo, serão descritos os objetivos gerais e específicos desta monografia, bem como a metodologia utilizada em seu desenvolvimento.

2.1 Objetivo Geral

O objetivo deste trabalho é avaliar o desempenho do protocolo de roteamento RPL em um cenário de alerta pós colisões veiculares.

2.2 Objetivos Específicos

A fim de cumprir o objetivo geral proposto, foram definidos os seguintes objetivos específicos:

- Revisão bibliográfica sobre VANET.
- Revisão bibliográfica sobre RPL.
- Revisão bibliográfica sobre o sistema operacional Contiki.
- Definição e implementação do modelo de disseminação de mensagens de alerta pós colisão veicular.
- Definição e implementação do cenário de testes no simulador Cooja.
- Variação dos parâmetros da função objetivo MRHOF e do algoritmo *trickle timer*.
- Análise dos resultados obtidos nas simulações e comparação com os requisitos impostos pela aplicação.

2.3 Materiais

Ao longo do desenvolvimento do trabalho foram utilizados os seguintes recursos: um computador para desenvolvimento dos códigos e execução das simulações, o emulador Cooja disponível na versão 4.9 do Contiki-NG para simulação dos cenários, o software Bonnmotion versão 3.0.1 para gerar os *traces* de mobilidade e a bibliografia sobre o tema para a realização das pesquisas.

2.4 Métodos

A primeira etapa deste trabalho consistiu da revisão bibliográfica dos conceitos fundamentais envolvendo VANET, RPL e o sistema operacional Contiki, e dos trabalhos relacionados a disseminação de mensagens de emergência em VANETs e ao uso do RPL em sistemas com movimento.

Em seguida, utilizando o Contiki-NG versão 4.9, foi desenvolvido o *firmware* para as RSUs (Road-Side Unit), que pode ser visto no apêndice A, e o *firmware* dos veículos, que consta no apêndice B, ambos seguindo a aplicação descrita no item 4.2. Após isso, através do

Bonnmotion, foram criados padrões de mobilidade baseados no modelo Manhattan Grid. Esses padrões foram projetados para simular cenários variados, especificamente com 85, 100, 200 e 300 veículos em movimento.

A fim de se avaliar o desempenho do protocolo, foram realizadas simulações computacionais utilizando o emulador Cooja, as quais foram executadas em duas etapas. Na primeira etapa, com o intuito de compreender como fatores externos ao protocolo podem afeta-lo, foram realizadas simulações variando o número de veículos (85, 100, 200 e 300), o alcance do rádio (100 m e 400 m) e o número de RSUs (entre 1 e 5). Na segunda etapa o foco recaiu sobre as configurações do protocolo, nela foram variados os valores do *trickle timer* e da histerese da função objetivo MRHOF, para ambas as métricas utilizadas: ETX (*Expected Transmission Count*) e RSSI (*Received Signal Strength Indication*). Para estas simulações, foram fixados o número de veículos em 150, o número de RSUs em 1 e o alcance do rádio em 100 m.

Com os resultados obtidos, foi realizada a análise do desempenho através de 3 principais métricas: PDR, atraso fim-a-fim e o alcance da disseminação das mensagens sendo elas comparadas com os requisitos impostos para a aplicação, conforme descrito no item 3.1.3. Além destas métricas, foram coletados dados a respeito da porcentagem de veículos que passaram pela área do acidente e foram notificados.

3 CONCEITOS FUNDAMENTAIS

Neste capítulo serão apresentados os principais conceitos necessários para a compreensão deste trabalho, sendo eles VANETs e sua aplicação em avisos de colisão, RPL e o sistema operacional Contiki.

3.1 VANET

As redes veiculares *Ad Hoc*, são classificadas como um subtipo de MANET (*Mobile Ad Hoc Network*) tendo como principal característica a presença de veículos como seus elementos móveis [6]. Este tipo de rede tornou-se crucial para o avanço dos ITS, já que ela permite a comunicação tanto entre veículos, denominada V2V (*Vehicle to vehicle*) como entre os veículos e infraestruturas instaladas ao longo das estradas, chamada de V2I (*Vehicle to Infrastructure*) [7]. Um dos grandes campos de pesquisa desta tecnologia está em seu uso para contribuir com a segurança rodoviária [8].

Essas redes possibilitam o desenvolvimento de aplicações que atuam na prevenção direta de acidentes, oferecendo recursos como avisos de colisão iminente, alertas de mudança de faixa e notificações de interseções, além de aplicações que auxiliam após um incidente, como é o caso dos alertas acionados após uma colisão [8]. Sendo o uso do RPL neste último caso citado o foco central de estudo deste trabalho.

Apesar de serem consideradas um tipo de MANET, as VANETs apresentam características próprias, decorrentes da presença de dispositivos em movimento em alta velocidade. Entre essas características, destaca-se a alta mobilidade, que resulta em atrasos nas comunicações V2V, a topologia altamente dinâmica, uma vez que os nós mudam de posição de forma constante e rápida, a relação crítica com o tempo e a volatilidade [9].

3.1.1 Arquitetura

De modo geral, a estrutura básica de uma VANET consiste de três diferentes tipos de dispositivos, OBUs (*On-Board Unit*), RSUs e TAs (*Trusted Authority*) [10]. As OBUs fazem coleta e processamento de dados nos veículos. Elas também são responsáveis por realizar a comunicação com as RSUs e com outros veículos [4]. As RSUs são os dispositivos instalados nas ruas, estradas e rodovias, permitindo que os veículos tenham acesso à internet, aumentando o alcance das VANETs e alertam possíveis incidentes e condições de trânsito [11]. Por fim, os TAs são os elementos responsáveis por gerenciar a rede como um todo, eles realizam a tarefa de autenticação e remoção de veículos e estão conectados via cabo com as RSUs, possuindo maior poder de processamento e armazenamento quando comparado com os outros dispositivos [4][11].

Nos últimos anos tem se popularizado cada vez mais o conceito de V2X (*Vehicle to Everything*), cujo objetivo é permitir que o veículo se comunique com qualquer dispositivo, dentro deste conceito destaca-se o V2V, V2I e V2P (*Vehicle to Pedestrian*) [12]. Cada uma dessas formas de comunicação pode contribuir para diferentes aplicações como por exemplo o V2V, que pode ser utilizado para alertas pré colisão, ou o V2I na implementação de alarmes gerados após incidentes e para controle tráfego e o V2P que pode ser utilizado para detecção de pedestres nas vias [13]. A Figura 3.1 ilustra a estrutura básica das VANETs, com seus diferentes elementos e tipos de comunicação.

Diversos protocolos de comunicação são estudados para uso nas VANETs, podendo ser categorizados de três formas: de longo alcance, como o WiMax (Worldwide Interoperability for Microwave Access) e LoRaWan (*Long Range Wide Area Network*), os de médio alcance, como o WIFI e o DRSC (*Dedicated Short-Range Communications*) e os de curto alcance que englobam o Bluetooth e o ZigBee [12][13].

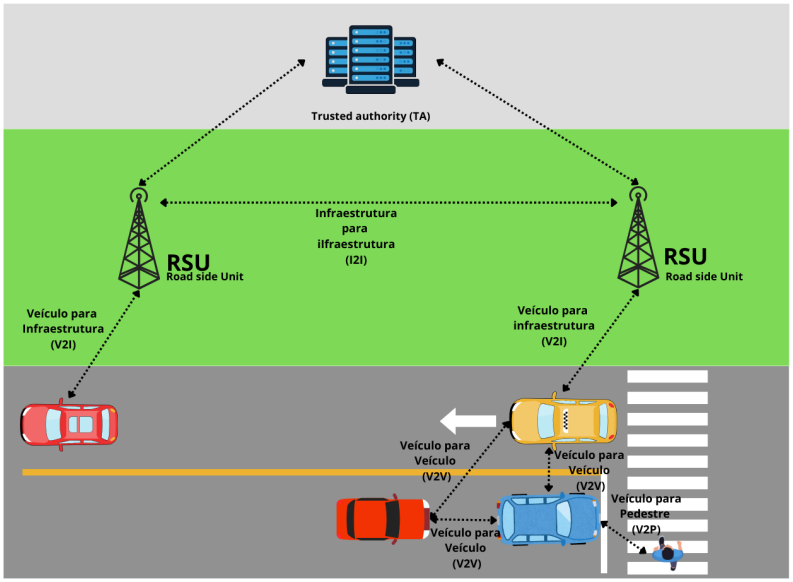


Figura 3.1: Topologia padrão das VANETs

3.1.2 Mensagens de Emergência

A transmissão de mensagens de emergência representa um grande desafio às VANETs, já que este tipo de rede possui características intrínsecas que propiciam o aumento do atraso e da perda de pacotes. Como essas mensagens visam alertar os motoristas sobre possíveis incidentes, é crucial que elas sejam entregues dentro de um tempo pré determinado, com o mínimo de perdas possível. Diversos trabalhos se propõe a desenvolver um método de transmissão dessas mensagens. Segue a descrição de alguns deles.

Rastogi et al.[14] propõe uma topologia centralizada nas eNBs (*Evolved Node B*) utilizando a tecnologia LTE (*Long Term Evolution*) e focando na comunicação V2I para a dispersão inicial das mensagens, empregando a comunicação V2V para aumentar o alcance do alerta. O fluxo do alerta se dá da seguinte forma: primeiro, o veículo que detecta o incidente envia esta informação para a eNB que está conectada, ela por sua vez, irá encaminhar para as eNBs vizinhas. Elas irão enviar uma mensagem *unicast* para todos os veículos conectados, com exceção do que originou o evento. Além disso, as eNBs criam *clusters* entre os veículos, definindo um como o *cluster head* (CH) e os demais como *cluster members* (CM), quando o CH recebe um *unicast* de emergência ele realiza um *broadcast* para os demais veículos, no intuito de garantir o envio da mensagem. O PDR obtido se aproximou dos 100%, obtendo um aumento de 2% em relação a outros métodos testados, o maior ganho está relacionado ao atraso que ficou entre 140 e 196 ms, representando uma diminuição de 23% na comparação com os outros métodos.

A partir de uma abordagem utilizando SDN (*Software Defined Network*), Zhu et al. [15], adotam uma topologia híbrida, utilizando tanto a comunicação V2I quanto V2V através do protocolo IEEE 802.11p. Assim que ocorre um incidente, o veículo encaminha a mensagem para a RSU mais próxima. A RSU encaminha a mensagem para um controlador, que determina a área na qual a mensagem deve ser retransmitida e as RSUs que devem iniciar essa transmissão. Com isso inicia-se um processo de *broadcast* através das infraestruturas e os veículos reencaminham a mensagem se acharem necessário, baseado no tipo de emergência que receberam. O principal ganho deste método está na cobertura das mensagens, ou seja, na taxa de veículos que foram notificados, mostrando aumentos de até 56,5%.

Outro método possível e que é abordado no trabalho de Benkergadh e Duvallet [16], é o de clusterização utilizando apenas comunicações V2V. O processo se inicia com os veículos realizando o *broadcast* de mensagens denominadas "*Hello*" a partir da qual eles obtem as informações dos vizinhos. Finalizando-se esta etapa, os nós utilizam os dados obtidos para calcular um valor de *fitness*, o qual é utilizado para formar o *cluster*. O *cluster head* é definido como sendo o nó com o menor *fitness* dentre os vizinhos. Em relação aos demais métodos testados no trabalho, a proposta dos autores apresentou um ganho de PDR de até 8%, uma diminuição no *overhead* de mensagens de 25% e um aumento na durabilidade do *cluster* formado.

3.1.3 Alerta Pós Colisão

O alerta pós colisão nada mais é do que um tipo de mensagem de emergência. Assim como as outras possui restrição de tempo, um alcance que deve atingir, um atraso máximo e um PDR (*Packet Delivery Ratio*) mínimo, que devem ser atingidos para garantir a sua eficiência. Para sua categoria, são estabelecidos um atraso fim-a-fim de 100 ms, um alcance de 300 metros, um PDR mínimo de 95% e com uma taxa de atualização de 1 Hz [17, 18, 19, 20].

3.2 RPL

Definido como padrão em 2012, O RPL é um o protocolo de roteamento IPV6 (*Internet Protocol Version 6*) padronizado pela IETF através da RFC 6550 para redes LLN (*Low Power and Lossy Network*) [21] utilizando o algoritmo vetor distância baseado em DODAGs (*Destination Oriented Directed Acyclic Graph*), na qual as conexões entre os nós são direcionados a um nó raiz, com uma instância RPL consistindo de uma ou mais DODAGs [22]. Essas estruturas são hierárquicas, com os nós estabelecendo entre si uma relação de pai e filho, na qual um nó elege outro como seu pai caso seja definido que ele está em uma posição melhor em relação ao raiz. Essa qualidade de posicionamento é denominada *ranking* e é definida através de uma função chamada função objetivo. A Figura 3.2 ilustra a estrutura de uma DODAG demonstrando seu sentido de comunicação.

O processo de formação da rede é iniciado com o raiz enviando uma mensagem do tipo DIO (DODAG *Information Object*), que carrega as informações relevantes referentes a DODAG como seu ID, o ID da instância a qual ela pertence, seu modo de operação e o ponto de código da função objetivo [23]. O nó raiz estabelece para si o *rank* 0 [24]. Ao receber um DIO, o nó o retransmite, atualizando o valor do *rank* com o valor obtido através da função objetivo. Além disso, o nó realiza o envio de uma mensagem chamada DAO (*Destination Advertisement Object*) para que seja definida a rota de *downward* [25]. O destino dessa mensagem pode variar de acordo com o modo de operação da DODAG. Se ela estiver operando no modo *storing*, cada nó irá internamente armazenar a tabela de rotas da rede, e a mensagem DAO será enviada

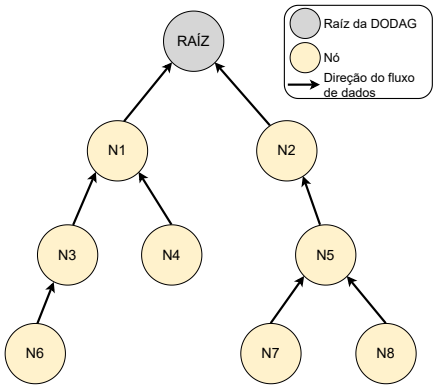


Figura 3.2: Topologia da DODAG.

ao pai selecionado. Já se ela estiver operando no modo *non-storing*, a tabela de rotas ficará armazenada apenas no raiz e o DAO será encaminhando sempre para ele [26].

Opcionalmente, é possível através da mensagem que seja solicitado um *ACK*, para que o nó tenha a confirmação da entrada na rede, sendo esta mensagem denominada DAO-ACK [27]. Esse processo está ilustrado na Figura 3.3.

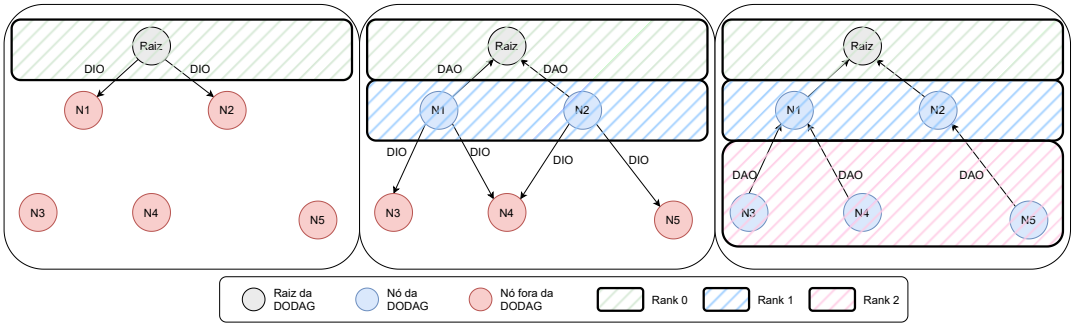


Figura 3.3: Formação da rede RPL com o fluxo de mensagens

Caso um nó, que já tenha selecionado o seu pai, receba uma mensagem DIO é feita uma verificação do *ranking* daquela mensagem, sendo comparado com o do pai atual e no caso dele ser menor a troca de parentesco é realizada. Após a formação da rede, ela é mantida através da transmissão das mensagens DIO que ocorrem seguindo a periodicidade determinada pelo algoritmo *trickle timer*, que será explicado mais à frente. Caso um nó deseje ingressar a uma rede já formada, ele pode disparar uma mensagem DIS (DODAG *Information Solicitation*), fazendo com que os nós que a recebem disparem o DIO com informações da DODAG.

3.2.1 Mensagens de Controle

Conforme mencionado anteriormente, a árvore de rotas é construída e mantida através das seguintes mensagens de controle: DIO, DAO, DAO-ACK e DIS [28], as quais serão mais detalhadas a seguir.

3.2.1.1 DIO

O DIO é a mensagem que carrega as informações necessárias para que um nó ingresse na DODAG, bem como as informações do nó que a enviou [29]. Ela é inicialmente enviada

pelo raiz para que a estrutura seja iniciada e é retransmitida pelos demais nós contendo suas informações específicas [23]. Dos diversos campos presentes na mensagem, vale-se destacar os seguintes [21]:

- RPLInstanceID: Campo de 8 bits preenchido pelo nó raiz, indica a qual instância RPL a DODAG pertence.
- Version Number: Campo de 8 bits preenchido pelo nó raiz. Ele define a versão da DODAG, só pode ser incrementado pelo nó raiz.
- Rank: Campo de 16 bits, indica o *rank* do nó dentro da DODAG.
- Mode of Operation (MOP): Campo de 3 bits que determina o modo de operação da instância RPL, podendo ser: Nenhuma rota descendente mantida pelo RPL; Modo de Operação Não-Armazenador; Modo de Operação Armazenador sem suporte a *multicast*; Modo de Operação Armazenador com suporte a *multicast*.
- DODAGPreference (Prf): Campo de 3 bits, define o *rank* da DODAG em relação a outras DODAGS da instância RPL.
- DODAGID: Campo de 128 bits, é um endereço *IPV6* setado pelo nó raiz, é utilizado para identificar a DODAG.
- OPTIONS: Um campo que pode ser utilizado para carregar informações a mais, sendo reponsável por armazenar o código que descreve a função objetivo.

3.2.1.2 DAO

O DAO é uma mensagem do tipo ascendente, ela é direcionada ao nó raiz caso a rede esteja operando em modo non *storing*, ou ao nó pai caso a mesma esteja operando no modo *storing* [30]. Essa mensagem é utilizada para formar as rotas de descidas da rede, a respeito dela vale-se destacar os seguintes campos [21]:

- RPLInstanceID: Campo de 8 bits preenchido pelo nó raiz, indica a qual instância RPL a DODAG pertence.
- K: *Flag* para sinalizar que o dispositivo que enviou a DAO aguarda uma confirmação.
- D: *Flag* para indicar a presença do campo DODAGID, sendo obrigatória no uso de uma RPLInstanceID local.
- DAOSequence: Campo de 8 bits, incrementado a cada mensagem do tipo DAO enviada pelo nó.
- DODAGID: Campo de 128 bits, é o identificador único da DODAG.

3.2.1.3 DIS

O DIS é a mensagem utilizada para solicitar informações a respeito de uma DODAG. Através dela um nó pode conhecer uma DODAG nova e ingressá-la [21]. Sua composição é mais simples, contendo três campos, sendo o mais importante o *options*, já que os outros dois, flags e *reserved*, devem ser zerados pelo transmissor e ignorados pelo receptor.

Através do *options* é sinalizado a requisição de informação, sendo possível que na solicitação sejam estabelecidos critérios que os nós devem cumprir antes de reiniciarem o *trickle timer* e encaminharem o DIO, como o ID da instância do RPL, o ID da DODAG e o número da versão.

3.2.1.4 DAO-ACK

A última mensagem de controle utilizada pelo protocolo é o DAO-ACK, ela é utilizada para confirmar o recebimento de um DAO. A resposta possui três categorias diferentes [21]:

- Sucesso (0): A solicitação foi aceita com sucesso.
- Rejeição indireta (1-127): Ocorre quando o nó aceita a condição de pai, contudo sugere que o solicitante procure uma rota alternativa.
- Rejeição (127-255): O nó rejeitou a solicitação.

Da mensagem são destacados os seguintes campos [21]:

- RPLInstanceID: Campo de 8 bits preenchido pelo nó raiz, indica a qual instância RPL a DODAG pertence.
- D: *Flag* para indicar a presença do campo DODAGID, sendo obrigatória no uso de uma RPLInstanceID local.
- DAOSequence: Campo de 8 bits, é a mesma DAOSequence da solicitação.
- Status: campo de 8 bits, é a resposta para a solicitação, sendo enquadrada em uma das categorias mencionadas anteriormente.
- DODAGID: Campo de 128 bits, é o identificador único da DODAG.

3.2.2 Trickle Timer

O *trickle timer* é um dos componentes responsáveis pela manutenção da rede, definido pela RFC 6206 [31], é o algoritmo responsável por controlar o disparo das mensagens DIO após a sua formação, regulando assim o fluxo de mensagens de controle [21]. A regulação é realizada através da supressão de mensagens e com a alteração exponencial do intervalo de envio [31]. O algoritmo utiliza como base três parâmetros: um intervalo mínimo I_{min}, um máximo I_{max} e uma constante de redundância k [32].

Seu funcionamento se dá da seguinte forma: Durante a inicialização, o intervalo I é configurado para ser o mesmo de I_{min} e o contador de inconsistências c é zerado. No começo de cada intervalo I, c é zerado e é definido um valor de tempo aleatório t, entre o intervalo $I/2$ e I [32]. Quando o tempo t é atingido a mensagem DIO é transmitida caso c seja menor que k. No momento em que I é atingido o seu valor é duplicado, sendo isto feito até que alcance I_{max}. Caso seja detectada alguma inconsistência, o valor de I retorna para I_{min} [33] e c para zero.

Em sua RFC, o *trickle* deixa a cargo do protocolo que o utiliza a definição das inconsistências, no caso do RPL a RFC 6550 define que um nó deve considerar inconsistências como sendo [21]:

- Quando há uma inconsistência no envio do pacote;

- O recebimento de um *multicast* DIS sem o campo de solicitação de informação, podendo ser desconsiderado caso haja uma *flag* restringindo o comportamento;
- O ingresso de um novo nó na rede;
- O recebimento de um *multicast* DIS com a opção de solicitação de informação no qual o nó preenche todos os requisitos estabelecidos na mensagem, podendo ser desconsiderado caso haja uma *flag* restringindo o comportamento.

3.2.3 Função Objetivo

Encarregada de estabelecer as métricas e os cálculos para a obtenção do *ranking* do nó [34], a função objetivo não se vincula a um algoritmo ou equação específicos definidos pela RFC. Essa flexibilidade permite sua adaptação a uma diversidade de cenários de uso [35]. A RFC 6551 delineia uma ampla gama de métricas, agrupando-as em duas categorias principais: métricas de *link* e métricas de nó [36].

As métricas de *link* abrangem aspectos relacionados à conexão entre os nós, tais como [37][9]:

- *Throughput*: Taxa média efetiva de um nó.
- Latência: É a latência daquele *link* expressa em microssegundos.
- ETX: Número esperado de transmissões para que um pacote seja entregue com sucesso.
- LQL: Medida da confiabilidade da conexão, indica o nível de qualidade do *link* expresso em valores de 0 a 7, sendo 0 indefinido, 1 o maior valor e 7 o menor.
- *LINK* color: É um valor de 10 bits, sendo a codificação de uma cor. Seu uso e significado ficam a cargo da aplicação

Quanto às métricas de nó, estas oferecem informações detalhadas sobre o dispositivo [37][9]:

- Estado do nó: Informações a respeito do sistema do nó, por exemplo métricas referentes a CPU, memória e carga de trabalho.
- Energia: Esse campo pode conter informações variadas a respeito da energia do nó, que podem ser o gasto para transmissão, energia restante, carga da fonte de alimentação, entre outros.
- Número de saltos: Número de saltos do nó até o raiz.

Apesar de seu uso não ser mandatório, foram padronizadas duas funções objetivo: a *Objective Function Zero* (OF0), conforme estabelecido pela RFC 6552 e a *Minimum Rank Hysteresis Objective Function* (MRHOF), conforme definido pela RFC 6719.

A OF0 foi a primeira proposta oficial de função objetivo. E se baseia no número de saltos do nó até o raiz e quanto menor esse número menor é o valor do *ranking* [38]. Logo, o caminho com menor número de saltos será sempre o melhor. Por utilizar apenas essa métrica, a função pode acabar gerando problemas de *link*, já que mesmo com um baixo número de saltos o nó pode não ter uma conexão confiável com o raiz, gerando retransmissões e perdas

desnecessárias de pacotes. Como essa métrica é estática e referente ao dispositivo, por mais que a condição da rota do ponto de vista do *link* esteja ruim, ela não é alterada já que o número de saltos se mantêm o mesmo [9].

Por conta desses problemas apresentados pela OF0, a MRHOF foi desenvolvida focando o seu funcionamento em uma métrica dinâmica de *link*, que é especificada na mensagem DIO. A função a utiliza como base para calcular o custo do caminho de todos os vizinhos, escolhendo inicialmente o de menor custo e definindo o seu *rank* a partir disto [39]. Além disso, existe um segundo mecanismo que é utilizado para a troca de parentesco, a qual só é realizada caso seja descoberto um novo caminho, cujo custo seja menor que o atual menos um valor de histerese [39]. A sua RFC não define qual métrica deve ser utilizada, porém determina que seja utilizada apenas uma e que ela deve ser aditiva, ou seja, seu valor é incrementado ao longo da rede.

Quando a métrica não é especificada, o MRHOF utiliza o ETX como padrão, o seu valor é calculado através da soma de dois parâmetros, o ETX entre o nó e o seu vizinho e o ETX total, que é o somatório dos valores de todas as conexões até o raiz [35]. Nessa implementação, o menor valor representa o melhor caminho e uma maior confiabilidade no *link*. A desvantagem desta abordagem é que por utilizar apenas uma métrica, outros parâmetros da rede acabam sendo ignorados, além disso os parâmetros físicos do nó também não são levados em consideração.

3.2.4 Impacto da Mobilidade

O RPL foi desenvolvido priorizando aplicações IoT (*Internet of Things*) estáticas, sem considerar a mobilidade dos nós, sendo este fator um dos grandes desafios enfrentados pelo protocolo [24]. A movimentação dos nós acarreta em uma diminuição do PDR, diminuição da qualidade de *link*, aumento na desconexão dos nós da rede e uma constante alteração da topologia o que faz com que mais mensagens de controle tenham que ser disparadas, por consequência aumentando o tráfego e fazendo com que os dispositivos consumam mais energia para tratá-las [40][41][42]. Em particular no caso das VANETs, esses problemas se agravam ainda mais, já que além de movimentos em direções distintas, esses cenários também podem apresentar uma alta densidade de nós e altas velocidades com a presença concomitante de dispositivos estáticos [43].

3.2.4.1 Árvore de Rotas

A Figura 3.4 demonstra como uma árvore de rotas pode ser afetada por um nó em movimento que a atravessa. Originalmente a instância contem 7 nós, sendo eles N1, N2, N3, N4, N5, N6 e o raiz. No primeiro instante, um nó em movimento denominado N7 se aproxima da rede, solicitando suas informações através de uma mensagem DIS, que é respondida por N2 e N4. A partir disso, N7 calcula seu *ranking* e o retransmite para os vizinhos, que fazem o mesmo, resultando na topologia vista no segundo instante, na qual N4 prefere se conectar a ele ao invés de manter a conexão direta com N2. No terceiro e último instante, N7 já se moveu o suficiente para estar fora do alcance de N4, que agora está sem conexão com a rede e sofrerá perdas de pacotes até conseguir reestabelecer sua rota com N2. Os nós N5 e N6 também sofrerão perdas, já que seus pacotes são transmitidos ao N4 que não possui mais rotas para acessar a rede.

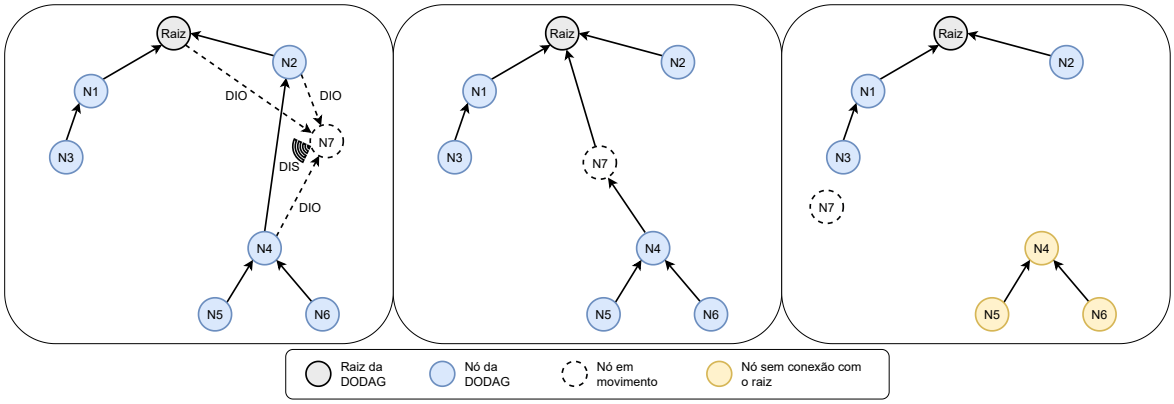


Figura 3.4: Alteração da árvore de rotas ocasionada por um nó em movimento

3.3 Trabalhos Relacionados

A fim de atenuar os impactos relacionados à mobilidade, diversos trabalhos foram desenvolvidos com diferentes propostas e focos de alterações. A seguir serão descritos os principais trabalhos desta área.

O trabalho desenvolvido por Corbazan et al. [41] visa reduzir os impactos no PDR causados pelos nós móveis aos estáticos. Para tal, é proposto um novo mecanismo de *trickle timer*, a identificação da mobilidade do nó através da mensagem DAO e a limitação da conexão dos nós móveis apenas como folhas. O maior cenário de teste foi realizado com 100 nós estáticos e 10 móveis, com uma velocidade de 1 m/s e direções aleatórias. O melhor resultado obtido foi um PDR de 9,81% contra 8,33% do protocolo padrão. Além disso ele obteve um menor atraso fim-a-fim.

Na abordagem adotada por Korbi et al. [42], os autores propõe novos mecanismos para tornar o RPL mais responsivo a alterações da topologia. Isso é realizado através da identificação da mobilidade do nó através de uma *flag* na mensagem DIO, alteração do algoritmo de seleção do pai, determinando que aqueles que não estão em movimento tenham preferência e o envio dinâmico de mensagens do tipo DIS, aumentando e diminuindo o seu intervalo conforme o número de trocas de pai realizados pelo nó. A avaliação da implementação foi realizada através da implementação de dois cenários, um apenas com movimentos lineares e outro com movimentos em malha. O PDR máximo atingido em ambos os cenários foi de cerca de 85% e um aumento de estabilidade nas rotas de até 20% em comparação com o RPL padrão. Não é informado o PDR do cenário utilizando o protocolo sem alterações, contudo há uma diminuição na perda de pacotes em relação a ele, também não é fornecida a velocidade de movimento, nem o *overhead* de mensagens de controle.

O mRPL [40] tem como base de seu funcionamento um mecanismo de *hand-off* denominado *smart-HOP*, que visa reduzir o tempo de troca de parentesco. Fica sob responsabilidade do pai identificar a movimentação do nó filho e solicitar que ele se desconecte e busque outro pai quando o ARSSi (*Average Received Signal Strength*) atingir um valor limite mínimo. Ao se desconectar, o nó faz o disparo de uma série de mensangens DIS e aguarda as respostas. Os testes foram realizados com nove nós estáticos e um nó se movimentando com uma velocidade de 2 m/s. Neste cenário, o mecanismo atingiu um PDR de quase 100% com atrasos menores que 95 ms, enquanto o RPL padrão atingiu um PDR de cerca de 50% não sendo informado o seu atraso. Sendo assim, o PDR apresentou um aumento relevante, mas não é possível avaliar outras métricas.

Em sua proposta Hoghooghi e Javidan [44], assim como no mRPL, visam reduzir os problemas decorrentes da desconexão dos nós. O que a difere dos outros trabalhos citados anteriormente, é que seu mecanismo não foca apenas em reagir as mudanças de topologia, mas sim detectar ativamente a mobilidade do nó, fazendo com que ele busque um novo pai antes de perder a conexão com o anterior, resultando na continuidade da conexão daquele dispositivo na rede. Essa detecção é realizada pelo pai, que solicita que o nó em movimento busque um novo pai, através de mensagens DIS. Os testes foram realizados com os nós movendo-se a até 2m/s, com esta velocidade consistindo no pior caso, a solução atingiu um PDR de 96,12% contra 89,26% do RPL padrão, já seu atraso fim-a-fim ficou em 231 ms contra os 359 ms do padrão, resultando em uma melhora tanto o atraso quanto o PDR.

O MARPL (*Mobility Aware* RPL), propõe um novo mecanismo de detecção de movimento e de detecção de indisponibilidade do nó pai [45]. Sua inovação em relação as abordagens citadas, está no fato de que a detecção de movimento é realizada através da variação do RSSI, além disso, ele utiliza um timer para monitorar se o pai está ou não fora de alcance. Quando o timer estoura, caso o nó não tenha recebido nenhuma mensagem dele nesse intervalo, então ele dispara mensagens DIS para procurar novos pais. Assim como o trabalho anterior, os testes foram realizados em baixas velocidades, indo até 3 m/s, contendo 30 nós móveis e um raiz. Neste cenário, o método atingiu um PDR de 22,13% e um atraso médio de 305,93 ms. Já o RPL padrão obteve 8,76% de PDR e um atraso de 251,21 ms. A solução trouxe um aumento significativo para a taxa de entrega, contudo resultou em maiores atrasos e mais mensagens de overhead.

Por fim o POF (*Programable Objective Function*) proposto por Fabian et Al. [43], utiliza lógica fuzzy para adaptar o uso das métricas de acordo com o cenário, podendo focar em apenas uma ou em um conjunto delas. Seu principal objetivo é aumentar a QoS (*Quality of Service*) em cenários IoV (*Internet of vehicles*). Os testes propostos foram os mais completos entre os avaliados, com quatro cenários distintos, nos quais foram variadas as velocidades máximas que atingiram até 54 km/h. Na velocidade mais alta, o PDR atingido foi de cerca de 14% com um atraso fim-a-fim de cerca de 400 ms. Em contra partida, a OF0 atinge 10% de PDR e 390 ms de atraso, já a MRHOF não atinge nem 1% de PDR e seu atraso fica por volta de 200ms. Apesar deste ganho na velocidade mais elevada, o resultado obtido em outras velocidades é muito semelhante ao da OF0, com o resultado inferior a ela em alguns casos.

3.4 Contiki-OS

O Contiki é um sistema operacional desenvolvido em C com foco em sistemas de baixa memória e poder de processamento, visando aplicações envolvendo redes de sensores sem fio [46]. O funcionamento do seu *kernel* é baseado em eventos, que são separados em duas categorias, os assíncronos, que são enfileirados para serem executados após um tempo e os síncronos, que são executados imediatamente. Junto do processamento dos eventos, o *kernel* também realiza um procedimento de *polling*, que roda entre a execução de processos assíncronos e é responsável por obter informações do hardware [47].

A nível de *kernel*, são implementadas apenas as funcionalidades mais básicas, o restante é realizado através de bibliotecas, que podem ser tanto as do núcleo do sistema, como também podem ser externas vinculadas à aplicação. Um dos exemplos mais relevantes de bibliotecas do núcleo é a que implementa a funcionalidade de *multithreading* com preempção, implementada através de uma pilha para cada *thread* criada [47].

Outro conceito importante para o Contiki é o de serviços, que consistem em processos para serem usados por outros processos. Sua vantagem é que podem ser alterados em

tempo de execução. Um exemplo de uso dos serviços está nas pilhas de comunicação, que foram implementadas desta forma, para que diversos protocolos possam ser implementados simultaneamente e alterados ao longo da execução dos programas [47].

4 AVALIAÇÃO DE DESEMPENHO

Neste capítulo serão descritos com mais detalhes a aplicação, o modelo de mobilidade, os cenários de testes implementados e serão discutidos os resultados obtidos

4.1 Aplicação

A aplicação para qual deseja-se estudar o uso do protocolo é o alarme pós colisão, ou seja, após dois veículos colidirem é emitida uma mensagem para os demais veículos alertando que naquela região houve um acidente. Este alarme possibilita que os motoristas fiquem atentos para alterar a sua rota, ou tomar cuidado ao passar pelo acidente.

Sua topologia foi baseada nos trabalhos de Rastogi et al. [14] e Wanting Zhu et al. [15], que centralizam o disparo das mensagens para os veículos da rede partindo de infraestruturas fixas. O primeiro motivo dessa escolha se deu pela característica do RPL de organizar a sua rede através de DODAGs, possuindo um nó central cujo papel pode ser desempenhado pelas RSUs. O segundo motivo foi por conta da facilidade de comunicação entre as estruturas, da mesma forma que ocorre nos trabalhos acima, assume-se que elas estão conectadas entre si. Por fim, por motivos de segurança, como mencionado ao longo da revisão a respeito das VANETs, de modo geral, as RSUs são conectadas com equipamentos capazes de autenticar os dispositivos contidos na rede, bem como removê-los, esse gerenciamento traz uma maior confiabilidade à rede.

O fluxo se dará da seguinte forma: o veículo que detectar o acidente será responsável por enviar ao nó raiz da rede, no caso uma RSU, a mensagem informando a colisão, contendo o horário em que ela ocorreu e informações das coordenadas do incidente realizando o envio a cada 1 segundo. Ao receber a mensagem, o raiz irá confirmar o recebimento ao veículo. A partir disso, a RSU irá retransmitir a mensagem para redes próximas que possam estar no alcance dos 300 metros requisitado pela aplicação, inserindo nela um ID identificando o acidente. Por fim a mensagem é retransmitida para todos os veículos dessas redes com exceção do que a originou. Um esquema ilustrando esta dinâmica pode ser visto na Figura 4.1

4.2 Modelo de Mobilidade

O padrão de movimentação é um dos pontos cruciais a serem estabelecidos em uma simulação com mobilidade, determinando o quanto ela se aproximará da realidade do ambiente que se deseja simular [48]. Os modelos de mobilidade podem ser categorizados em dois grupos principais: os modelos baseados em *traces*, nos quais os dados são coletados de cenários reais, e os modelos sintéticos, nos quais a movimentação dos nós é obtida por meio de funções matemáticas, algoritmos e física do movimento [48].

Os modelos sintéticos podem ser subdivididos em duas categorias: a de mobilidade baseada em grupo e a mobilidade baseada em entidade [49]. Na primeira, a movimentação de um nó sofre influência dos demais e dentro desta classificação estão presentes modelos como Nomadic Community, Reference Point Group [50]. Já na segunda a movimentação do nó depende apenas dele, sem sofrer interferências de outros elementos, podendo ser citados os modelos Random Way Point, Random Walk e Random Direction [51].

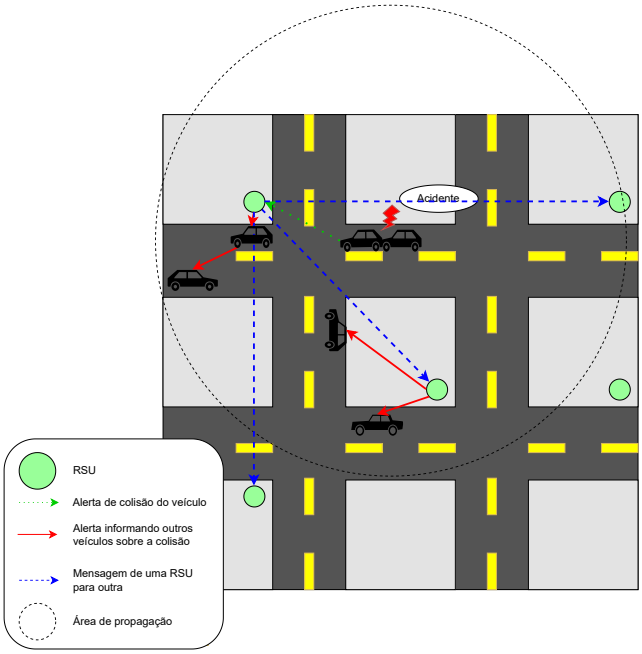


Figura 4.1: Aplicação de alarme de colisão.

4.2.1 Modelo de Mobilidade *Manhattan Grid*

O modelo *Manhattan Grid*, escolhido para este trabalho, foi inicialmente proposto por Safaei et al. [52], sendo composto por um mapa formado por ruas horizontais e verticais, nas quais os nós trafegam em dois sentidos [52]. Ao chegar em cada interseção, o nó conta com uma probabilidade de 50% de seguir em frente, 25% de virar à esquerda e 25% de virar à direita [53]. Sua velocidade pode variar ao longo de sua trajetória tendo o novo valor de velocidade uma dependência do valor anterior [54].

A mudança de velocidade é realizada a cada intervalo de distância percorrida e possui uma probabilidade de acontecer [55]. Na implementação realizada pelo Bonnmotion, é utilizado um intervalo de 5 metros, com uma probabilidade de alteração de 20%. A nova velocidade consiste da velocidade média atual acrescida de um valor aleatório que está dentro de uma distribuição gaussiana com média 0 e desvio padrão 1, multiplicado por uma constante de 0,2.

O número de quadras e seus tamanhos podem variar para se encaixar no tamanho de mapa desejado, no caso deste trabalho, foi definido um mapa com 100 quadras, sendo 10 na horizontal e 10 na vertical. O seu tamanho foi definido como sendo 100 metros por 100 metros, resultando em um mapa de 1000 metros por 1000 metros.

Este modelo se mostra adequado para simular a mobilidade em áreas urbanas, visto que a movimentação dos nós assemelha-se ao que é observado nas ruas de uma cidade [56]. O mapa gerado para a simulação pode ser visto na Figura 4.2

4.3 Cenários de Testes

Os testes foram definidos de modo a analisar o impacto de diferentes fatores no desempenho do RPL na aplicação e a performance das duas principais funções objetivo: a OF0 e a MRHOF com métrica de ETX e com métrica de RSSI. O mapa, o modelo de perda, o tipo de rádio utilizado e a faixa de velocidade dos veículos foram constantes em todos os cenários.

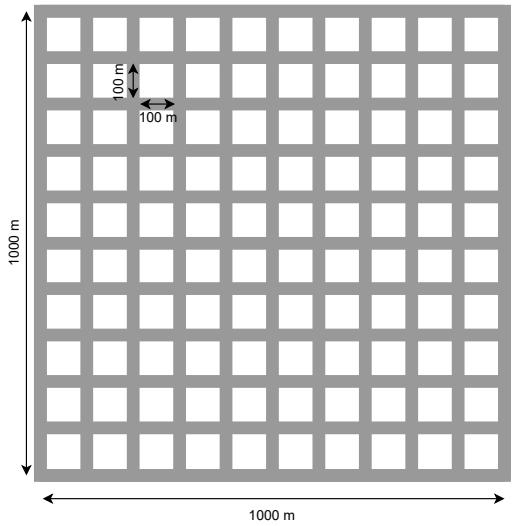


Figura 4.2: Mapa das simulações.

O modelo de perda utilizado é o UDGM (*Unit Disk Graph Medium*) disponível no emulador Cooja. Neste modelo a área de alcance do rádio é definida através de um círculo centralizado no nó, cujo raio é o alcance máximo do rádio.

Os nós localizados no interior do círculo possuem uma probabilidade de receber os pacotes enviados, essa probabilidade diminui conforme o receptor se aproxima da borda do círculo [57]. O seu valor é definido com um cálculo realizado com base em três valores, a distância entre os nós d, o alcance máximo d_{max} e a taxa de sucesso de recepção P_{Rx}, sendo esta definida como padrão como sendo 100% pelo Cooja, podendo ser alterada na simulação. A Equação 4.1 demonstra o cálculo realizado para obter a probabilidade.

$$P = 1 - \frac{d^2}{d_{max}^2}.(1 - P_{Rx}) \tag{4.1}$$

O rádio utilizado é o IEEE 802.15.4, operando na frequência de 2.4 GHz. Nesta frequência, o IEEE 802.15.4 conta com 16 canais, cada um contendo uma largura de banda de 2 MHz e um espaçamento de 5 MHz entre eles [58][59]. Por fim, a velocidade máxima dos veículos variaram entre 10 km/h até 80 km/h,

Conforme mencionado no item 2, os testes foram realizados em duas etapas distintas. Na primeira, foram variados apenas os valores externos ao protocolo, como o número de veículos, utilizando os valores de 85, 150, 200 e 300, o número de RSUs variando entre 1 e 5 e o alcance do rádio que variou entre 100 m e 400 m. Os valores adotados podem ser vistos com mais detalhes na Tabela 4.1.

Foi necessário criar uma topologia de posicionamento para cada número de RSUs utilizado. As Figuras 4.3 e 4.4 ilustram o mapa com 1 RSU e 5 RSUs, respectivamente.

Já na segunda etapa foram variados os valores de *trickle timer* e a histerese das métricas utilizadas na função objetivo MRHOF. Para estas simulações foi definido o número de veículos como sendo 150, foi utilizada apenas uma RSU e a distância de alcance dos rádios foi definida como 100 m de acordo com a limitação da tecnologia [60]. Todos os valores podem ser vistos na Tabela 4.2.

Parâmetros gerais	
Parâmetros	Valor
Emulador	Cooja
Tamanho do mapa	1000 m x 1000 m
Protocolo	IEEE 802.15.4
Frequência de operação	2.4 Ghz
Número de canais	16
Largura de banda dos canais	2 Mhz
Alcance dos rádios	[100, 400] m
Número de veículos	[85, 150, 200, 300]
Número de RSUs	[1, 5]
Velocidade máxima dos veículos	[10-80] Km/h
Ambiente	Urbano
Modelo de perda	UDGM
Função Objetivo	[OF0, MRHOF]
Intervalo mínimo do Tickle timer	512 ms
Intervalo máximo do Tickle timer	4,096 s
Constante de redundância	10
Tempo de armazenamento de rota	10 s
Parâmetros MRHOF ETX	
Parâmetros	Valor
Métricade link	ETX
Threshold para troca de pai	96
Custo máximo do link	1024
Custo máximo do caminho	32768
Parâmetros MRHOF RSSI	
Parâmetros	Valor
Métricade link	RSSI
Threshold para troca de pai	15 dB
Custo máximo do link	85 dB
Custo máximo do caminho	290

Tabela 4.1: Parâmetros das simulações da primeira etapa.

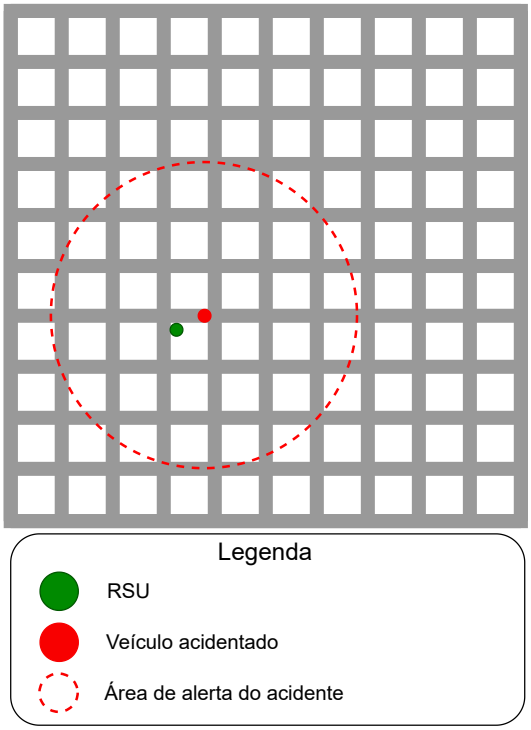


Figura 4.3: Topologia 1 RSU.

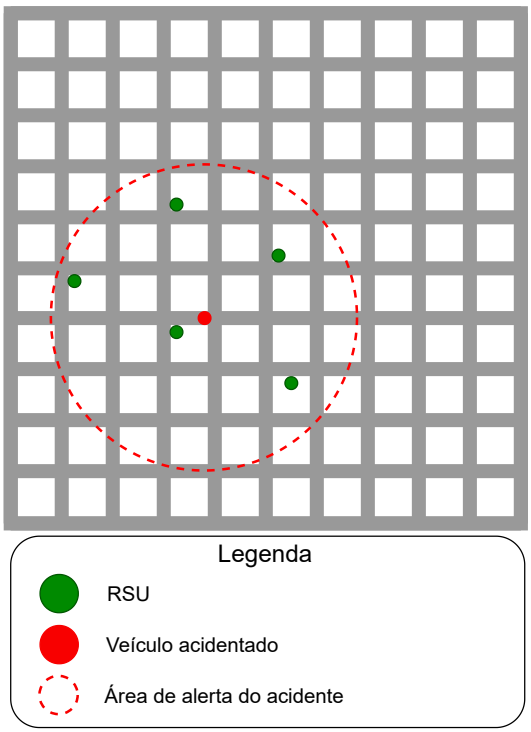


Figura 4.4: Topologia 5 RSUs.

Parâmetros gerais	
Parâmetros	Valor
Emulador	Cooja
Tamanho do mapa	1000 m x 1000 m
Protocolo	IEEE 802.15.4
Frequência de operação	2.4 Ghz
Número de canais	16
Largura de banda dos canais	2 Mhz
Alcance dos rádios	100 m
Número de veículos	150
Número de RSUs	1
Velocidade máxima dos veículos	[10-80] Km/h
Ambiente	Urbano
Modelo de perda	UDGM
Função Objetivo	[OF0, MRHOF]
Intervalo mínimo do Tickle timer	[8 ms - 4,096 s]
Intervalo máximo do Tickle timer	4,096 s
Constante de redundância	10
Tempo de armazenamento de rota	10 s
Parâmetros MRHOF ETX	
Parâmetros	Valor
Métricade link	ETX
Threshold para troca de pai	[1 - 1024]
Custo máximo do link	1024
Custo máximo do caminho	32768
Parâmetros MRHOF RSSI	
Parâmetros	Valor
Métricade link	RSSI
Threshold para troca de pai	[1 - 85] dB
Custo máximo do link	85 dB
Custo máximo do caminho	290

Tabela 4.2: Parâmetros das simulações da segunda etapa.

4.3.1 Primeira Etapa de Testes

Conforme mencionado anteriormente, os testes foram realizados em duas etapas, na primeira foram variados apenas fatores externos ao protocolo. A Tabela 4.3 apresenta os resultados obtidos para as funções objetivo OF0, MRHOF com métrica de ETX e MRHOF com métrica de RSSI. Ela está organizada da seguinte forma: as três primeiras colunas correspondem aos parâmetros das simulações, em seguida estão presentes os resultados para as duas funções testadas, apresentando o PDR, atraso, número de carros notificados (NCN), número de carros notificados em porcentagem e distância média do primeiro aviso (DMPA)

Parâmetros			OF0					MRHOF – ETX					MRHOF – RSSI				
RSUs	Alcance rádio (m)	NC	PDR (%)	Atraso (ms)	NCN	NCN (%)	DMPA (m)	PDR (%)	Atraso (ms)	NCN	NCN (%)	DMPA (m)	PDR (%)	Atraso (ms)	NCN	NCN (%)	DMPA (m)
1	100	85	24,14%	75,40	44	51,76%	81,21	24,14%	75,40	44	51,76%	76,85	21,30%	55,23	24	28,24%	72,41
		150	11,86%	86,38	74	49,33%	85,35	11,74%	82,69	45	30,00%	70,87	24,92%	58,51	45	30,00%	58,8
		200	16,92%	140,56	108	54,00%	103,92	14,75%	116,21	41	20,50%	80,38	28,88%	65,25	36	18,00%	64,03
		300	11,34%	154,03	141	47,00%	84,24	12,30%	110,63	60	20,00%	88,8	17,64%	62,99	50	16,67%	60,27
	400	85	46,07%	603,38	85	100,00%	436,71	40,21%	2.044,84	85	100,00%	436,78	76,65%	72,71	85	100,00%	397,19
		150	10,95%	3.016,88	142	94,67%	401,72	24,01%	2.405,77	148	98,67%	396,95	63,31%	93,45	148	98,67%	370,69
		200	3,57%	5.424,16	101	50,50%	339,16	11,30%	4.100,25	149	74,50%	342,8	31,49%	322,69	182	91,00%	351,51
		300	1,76%	8.149,65	46	15,33%	231,2	2,47%	15.916,29	42	14,00%	270,67	22,62%	394,99	213	71,00%	277,66
5	100	85	14,58%	52,97	51	60,00%	232,76	14,45%	52,97	51	60,00%	232,77	19,38%	51,68	36	42,35%	216,57
		150	12,77%	65,18	75	50,00%	237,68	11,88%	56,40	51	34,00%	228,8	25,13%	52,50	69	46,00%	242,97
		200	15,07%	70,71	115	57,50%	251,21	15,49%	63,88	63	31,50%	241,1	25,22%	50,80	79	39,50%	217,41
		300	14,32%	98,98	170	56,67%	252,73	15,29%	64,21	113	37,67%	231,93	26,51%	64,48	117	39,00%	234,4
	400	85	49,93%	605,31	85	100,00%	443,69	54,01%	708,60	85	100,00%	449,56	70,54%	75,38	85	100,00%	434,76
		150	8,13%	8.022,37	143	95,33%	378,58	22,81%	5.877,56	150	100,00%	423,64	57,42%	150,43	150	100,00%	374,52
		200	5,75%	16.099,66	113	56,50%	399,8	13,82%	5.036,56	195	97,50%	439,48	43,22%	213,25	199	99,50%	393,94
		300	2,50%	10.552,07	28	9,33%	309,57	5,72%	11.548,19	124	41,33%	356,017	16,73%	1.950,39	203	67,67%	354,98

Tabela 4.3: Resultados das simulações variando parâmetros externos

O primeiro ponto que é possível perceber através dos resultados é que esses componentes externos têm um grande impacto na aplicação. Em cenários com alcances menores, a OF0 obteve melhores resultados em relação ao número de carros notificados e à distância da primeira notificação. Contudo, seu atraso acabou sendo o pior dos três, ficando fora dos limites impostos pela aplicação nos testes com uma RSU, tanto com 200 carros quanto com 300. Neste último quesito, a MRHOF com métrica de RSSI se destacou, obtendo valores de atraso menores que 100 ms em todos os testes cujo alcance do rádio foi de 100 m. O aumento no número de RSUs nestes cenários trouxe um aumento tanto no número de carros notificados quanto na distância média do primeiro aviso. Houve também uma melhora no atraso para a OF0 e para a MRHOF com ETX, o que não ocorreu para a MRHOF com RSSI.

Já nos cenários com um alcance maior, a MRHOF com RSSI obteve melhores valores em todas as métricas, principalmente no atraso fim-a-fim, que se manteve abaixo em comparação às demais funções em todos os testes. Nos testes com 200 e 300 veículos, houve um aumento significativo no atraso e uma queda no PDR e no número de veículos notificados, o que demonstra uma dificuldade para a rede convergir, devido ao alto número de dispositivos tentando ingressar na mesma DODAG. O mesmo problema ocorreu de forma muito mais acentuada para a MRHOF com ETX e para a OF0, que chegaram a ter um atraso de 15 e 8 segundos e um PDR de 2,47% e 1,76%, respectivamente. Aumentar o número de RSUs neste caso não foi tão benéfico quanto no caso anterior; de maneira geral, houve um pequeno aumento no PDR e no número de veículos notificados, contudo, todas as funções sofreram um aumento significativo no atraso.

É importante ressaltar que os resultados se alteram de maneira expressiva para os diferentes cenários testados, com o PDR variando de 1,76% até 76,65%, o atraso variando de 50,8 ms até 15,916 s e a porcentagem de carros notificados indo de 1,76% a 100%. Ainda assim, é possível observar que cada fator acaba impactando de forma semelhante as funções. Por exemplo, aumentar o número de veículos quase sempre diminui o PDR e aumenta o atraso, principalmente para alcances maiores de rádio, justamente porque a rede tem mais dificuldade em convergir, já que cada dispositivo que entra gera uma atualização nela que pode impactar os que já estão presentes. Por sua vez, aumentar o número de RSUs pode melhorar o PDR e,

para alcances menores, o atraso, pois elas possibilitam que os dispositivos fiquem divididos em mais redes. Por fim, alcances maiores de rádio tendem a fazer com que mais veículos sejam notificados, mas como mencionado anteriormente, se o número de veículos for muito alto, isso se torna um problema para a convergência da rede.

Outro ponto que acaba sendo evidenciado pelos resultados é como cada função se comporta diante dessa variação. A OF0 foi a que possibilitou que mais veículos fossem notificados quando o alcance do rádio era menor. Os veículos possuem mais facilidade para entrar na rede nesses casos, porém, não há um cuidado com a qualidade do link destes dispositivos, o que se reflete no PDR e no atraso. O MRHOF com ETX sofreu de algo semelhante, mas notificou menos veículos que a OF0. Já quando a métrica de RSSI foi utilizada, a qualidade dos links das rotas estabelecidas acabou sendo muito melhor, o que é visível pelo atraso e pelo PDR, principalmente com alcances maiores do rádio. Entretanto, a rede com esta métrica não se mostrou muito adaptável, o que é evidenciado quando se observam os cenários com um alcance menor de rádio.

4.3.2 Segunda Etapa de Testes

Na segunda etapa foram variados os valores referentes a configuração da função objetivo e do *trickle timer*, partindo da função objetivo, o parâmetro variado foi a histerese de cada métrica da MRHOF. Para o RSSI, o seu valor de histerese foi variado de 1 dB até 85 dB, sendo obtido o gráfico da Figura 4.5

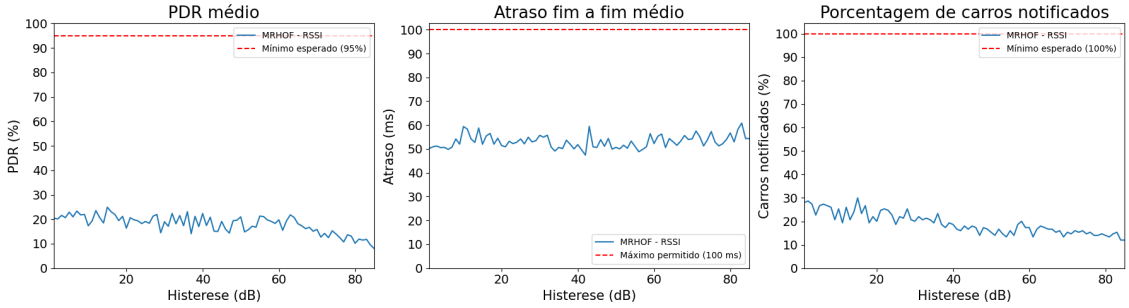


Figura 4.5: Resultados com a variação da histerese de RSSI.

Em seguida o mesmo foi feito para o ETX, variando o valor de histerese de 0,0078 até 8, com resultados mostrados na Figura 4.6

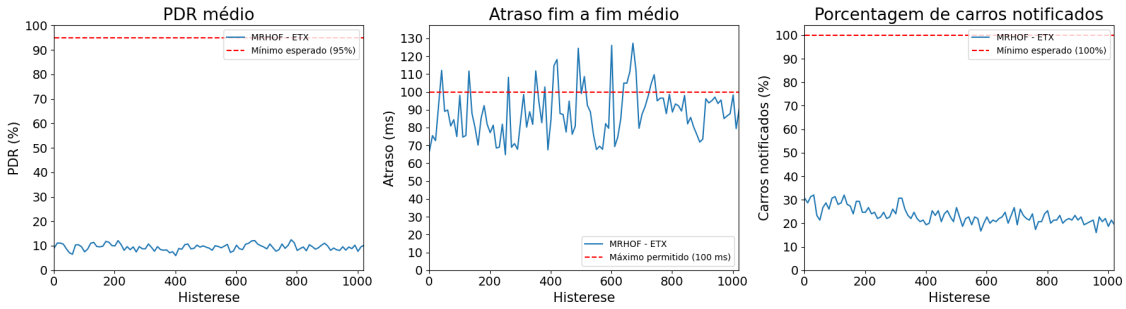


Figura 4.6: Resultados com a variação da histerese de ETX.

Os gráficos evidenciam o impacto da histerese nos resultados do desempenho do protocolo. Em ambas as métricas, há uma tendência de diminuição no número de carros notificados à medida que a histerese aumenta. Isso ocorre, pois esse aumento faz com que a

rede tenda a sofrer menos alterações, já que os nós trocarão menos pais. No caso do RSSI, essa queda é muito mais drástica do que no caso do ETX. Contudo, o atraso fim-a-fim e o PDR do RSSI se mostraram melhores na maioria dos casos. Inclusive, em nenhum momento o atraso passou dos 100 ms, ao contrário do ETX, que por diversas vezes acabou excedendo esse valor.

Por fim, as funções foram submetidas à variação do valor mínimo do *trickle timer*, variando de 4 ms até 16,384 s, calculado conforme mostrado a seguir

$$I_{min} = 2^n \tag{4.2}$$

onde n é o expoente configurado no teste e I_{min} é o tempo dado em ms. Os gráficos obtidos estão em função do expoente n, que varia de 2 até 14. A Figura 4.7 demonstra, respectivamente, os Gráficos de PDR, atraso fim-a-fim e de números de carros notificados em relação à variação do valor do temporizador para as funções objetivo OF0, MRHOF com métrica de ETX e MRHOF com métrica de RSSI.

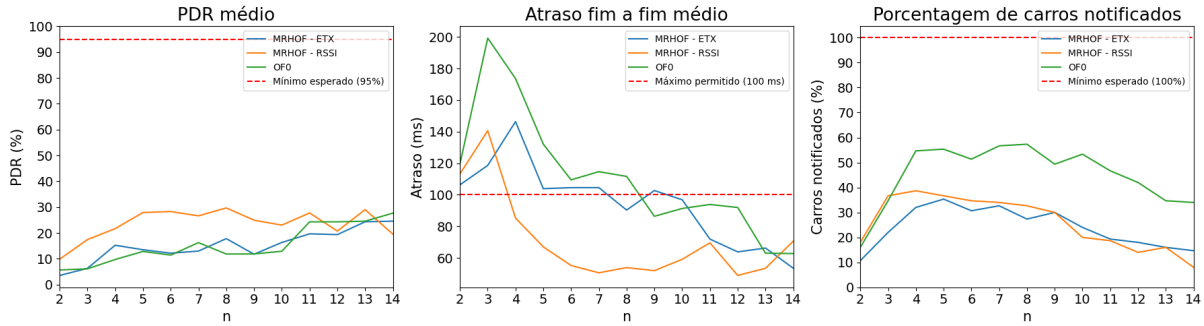


Figura 4.7: Resultados com a variação do *trickle timer*

Assim como o valor de histerese para o MRHOF, o valor do *trickle timer* altera diretamente o desempenho do RPL. Valores mais baixos tendem a aumentar o atraso, enquanto o PDR tende a diminuir, o que é esperado, visto que o *trickle timer* é responsável por controlar o disparo das mensagens que detectam alterações na rede. Como essas alterações podem gerar mudanças nas rotas, esse envio excessivo de mensagens faz com que a rede fique em constante mudança, dificultando a convergência da mesma.

Em relação ao número de carros notificados, este acaba sofrendo tanto com valores muito baixos quanto com valores muito altos. Isso ocorre por uma lógica semelhante à anterior: valores baixos demais dificultam a convergência da rede, fazendo com que os carros não sejam notificados, enquanto valores muito altos fazem com que os carros não consigam ingressar na rede, também não sendo notificados.

A OF0 foi a função que conseguiu atingir o maior número de veículos, mas, em compensação, resultou em maiores valores de atraso, assim como ocorreu na primeira etapa de testes. Nesta configuração de histerese, o RSSI obteve um número maior de veículos notificados do que o ETX, ficando atrás apenas da OF0. Além disso, o MRHOF com RSSI apresentou os melhores valores de PDR e atraso, com este último estando abaixo dos 100 ms na maior parte do tempo.

Aqui novamente fica evidente a importância das configurações para o desempenho do protocolo. A partir de uma mudança no *trickle timer*, foi possível fazer com que uma função que antes tinha um resultado pior do que a outra passasse a ter um desempenho melhor, como é o caso do RSSI em relação ao ETX.

5 CONCLUSÕES E TRABALHOS FUTUROS

O RPL, com suas funções objetivo padrão, não foi concebido para lidar com sistemas que apresentam mobilidade, como evidenciado pelos resultados discutidos na seção anterior. A principal dificuldade reside na capacidade da rede em manter a convergência mesmo diante de uma alta mobilidade dos nós. Enquanto os nós que transitam rapidamente pela área de alerta precisam ser integrados à rede e informados sobre o acidente sem demora, eles não devem comprometer a comunicação dos nós já estabelecidos. Isso requer que o protocolo seja ágil na detecção e tratamento de mudanças na topologia da rede. O protocolo não foi capaz de satisfazer os requisitos mínimos da aplicação, que exige um PDR de 95%, um atraso máximo de 100 ms e que todos os veículos em uma área de 300 metros sejam alertados. O desempenho do RPL é, de maneira positiva e negativa, fortemente influenciado pelas configurações do *trickle timer* e da função objetivo. A simulações demonstram, que faixas de valor menores do *trickle timer* tendem a possibilitar que mais veículos sejam notificados, porém se forem baixos demais a rede acaba tendo dificuldades para se formar devido a constante atualização, fazendo com que o número de nós notificados reduza. Outro ponto relevante está no fato de que as funções objetivos tiveram diferenças entre si e uma mesma função com métricas distintas também obteve resultados diferentes. É importante notar que a alteração dos parâmetros do RPL original resultou em ganhos ganhos comparáveis ou até maiores em alguns casos, quando comparados com as propostas de alteração do protocolo para suportar redes móveis. Desta forma, a adaptação do RPL à situações de mobilidade permanece um tema de pesquisa em aberto.

REFERÊNCIAS

[1] A. Ghandi and S. Paltsev, “Global co2 impacts of light-duty electric vehicles,” *Transportation Research Part D: Transport and Environment*, vol. 87, p. 102524, 2020.

[2] WHO, “Global status report on road safety 2018,” Tech. Rep. ISBN 9789241565684, World Health Organization, Geneva, 2018.

[3] M. N. Mejri, J. Ben-Othman, and M. Hamdi, “Survey on vanet security challenges and possible cryptographic solutions,” *Vehicular Communications*, vol. 1, no. 2, pp. 53–66, 2014.

[4] M. S. Sheikh, J. Liang, and M. A. Khan, “A comprehensive survey on vanet security services in traffic management system,” *Wirel. Commun. Mob. Comput.*, vol. 2019, jan 2019.

[5] T. Yeferny and S. Hamad, “Vehicular ad-hoc networks: Architecture, applications and challenges,” 2021.

[6] M. A. Karabulut, A. F. M. S. Shah, H. Ilhan, A.-S. K. Pathan, and M. Atiquzzaman, “Inspecting vanet with various critical aspects – a systematic review,” *Ad Hoc Networks*, vol. 150, p. 103281, 2023.

[7] T. Karunathilake and A. Förster, “A survey on mobile road side units in vanets,” *Vehicles*, vol. 4, no. 2, pp. 482–500, 2022.

[8] M. Maad Hamdi, L. Audah, S. Abduljabbar Rashid, A. Hamid Mohammed, S. Alani, and A. Shamil Mustafa, “A review of applications, characteristics and challenges in vehicular ad hoc networks (vanets),” in *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pp. 1–7, 2020.

[9] A. Brachman, “RPL objective function impact on llns topology and performance,” in *Internet of Things, Smart Spaces, and Next Generation Networking* (S. Balandin, S. Andreev, and Y. Koucheryavy, eds.), (Berlin, Heidelberg), pp. 340–351, Springer Berlin Heidelberg, 2013.

[10] F. Azam, S. K. Yadav, N. Priyadarshi, S. Padmanaban, and R. C. Bansal, “A comprehensive review of authentication schemes in vehicular ad-hoc network,” *IEEE Access*, vol. 9, pp. 31309–31321, 2021.

[11] P. Mundhe, S. Verma, and S. Venkatesan, “A comprehensive survey on authentication and privacy-preserving schemes in vanets,” *Computer Science Review*, vol. 41, p. 100411, 2021.

[12] S. Sharma, A. Kaul, S. Ahmed, and S. Sharma, “A detailed tutorial survey on vanets: Emerging architectures, applications, security issues, and solutions,” *International Journal of Communication Systems*, vol. 34, no. 14, p. e4905, 2021.

[13] K. F. Haque, A. Abdelgawad, V. P. Yanambaka, and K. Yelamarthi, “Lora architecture for v2x communication: An experimental evaluation with vehicles on the move,” *Sensors*, vol. 20, no. 23, 2020.

[14] E. Rastogi, M. K. Maheshwari, A. Roy, N. Saxena, and D. R. Shin, “A novel safety message dissemination framework in lte-v2x system,” *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 9, p. e4275, 2021.

[15] W. Z. H. Z. Wanting Zhu, Deyun Gao and H.-P. Chiang, “Sdn-enabled hybrid emergency message transmission architecture in internet-of-vehicles,” *Enterprise Information Systems*, vol. 12, no. 4, pp. 471–491, 2018.

[16] S. Benkerdagh and C. Duvallet, “Cluster-based emergency message dissemination strategy for vanet using v2v communication,” *International Journal of Communication Systems*, vol. 32, no. 5, p. e3897, 2019. e3897 dac.3897.

[17] Q. Xu, T. Mak, J. Ko, and R. Sengupta, “Vehicle-to-vehicle safety messaging in DSRC,” in *Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks*, VANET ’04, (New York, NY, USA), p. 19–28, Association for Computing Machinery, 2004.

[18] V. Nguyen, O. T. T. Kim, C. Pham, T. Z. Oo, N. H. Tran, C. S. Hong, and E.-N. Huh, “A survey on adaptive multi-channel mac protocols in vanets using markov models,” *IEEE Access*, vol. 6, pp. 16493–16514, 2018.

[19] A. T. Sasongko, G. Jati, B. Hardian, and W. Jatmiko, “The reliability of routing protocols as an important factor for road safety applications in vanet-based autonomous cars,” *Journal of Computer Science*, vol. 16, pp. 768–783, Jun 2020.

[20] “Vehicle safety communications project task 3 final report: Identify intelligent vehicle safety applications enabled by DSRC,” Mar 2005. Tech Report.

[21] R. Alexander, A. Brandt, J. Vasseur, J. Hui, K. Pister, P. Thubert, P. Levis, R. Struik, R. Kelsey, and T. Winter, “RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks.” RFC 6550, Mar. 2012.

[22] F. ARAT and S. DEMİRCİ, “Experimental analysis of energy efficient and qos aware objective functions for RPL algorithm in iot networks,” *Sakarya University Journal of Computer and Information Sciences*, vol. 4, pp. 192–203, 08 2021.

[23] O. Gaddour, A. Koubâa, S. Chaudhry, M. Tezeghdanti, R. Chaari, and M. Abid, “Simulation and performance evaluation of dag construction with RPL,” in *Third International Conference on Communications and Networking*, pp. 1–8, 2012.

[24] H.-S. Kim, J. Ko, D. E. Culler, and J. Paek, “Challenging the ipv6 routing protocol for low-power and lossy networks (RPL): A survey,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2502–2525, 2017.

[25] F. Gara, L. Ben Saad, E. Ben Hamida, B. Tourancheau, and R. Ben Ayed, “An adaptive timer for RPL to handle mobility in wireless sensor networks,” in *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 678–683, 2016.

[26] B. Ghaleb, A. Al-Dubai, E. Ekonomou, M. Qasem, I. Romdhani, and L. Mackenzie, “Addressing the dao insider attack in RPL’s internet of things networks,” *IEEE Communications Letters*, vol. 23, no. 1, pp. 68–71, 2019.

[27] A. E. Hassani, A. Sahel, and A. Badri, “Assessment of a proactive routing protocol RPL in ipv6 based wireless sensor networks,” in *2019 Third International Conference on Intelligent Computing in Data Sciences (ICDS)*, pp. 1–7, 2019.

[28] F. Medjek, D. Tandjaoui, N. Djedjig, and I. Romdhani, “Multicast dis attack mitigation in RPL-based iot-llns,” *Journal of Information Security and Applications*, vol. 61, p. 102939, 2021.

[29] N. Accettura, L. A. Grieco, G. Boggia, and P. Camarda, “Performance analysis of the RPL routing protocol,” in *2011 IEEE International Conference on Mechatronics*, pp. 767–772, 2011.

[30] J. Rodrigues Cotrim, E. Arata, and J. Kleinschmidt, *Roteamento para Internet das Coisas - Protocolos, Mobilidade e Segurança*. 2017.

[31] A. Musaddiq, Y. B. Zikria, and S. W. Kim, “Energy-aware adaptive trickle timer algorithm for RPL-based routing in the internet of things,” in *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, pp. 1–6, 2018.

[32] P. Levis, T. H. Clausen, O. Gnawali, J. Hui, and J. Ko, “The Trickle Algorithm.” RFC 6206, Mar. 2011.

[33] K. C. Lee, R. Sudhaakar, L. Dai, S. Addepalli, and M. Gerla, “RPL under mobility,” in *2012 IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 300–304, 2012.

[34] V. C. Diniesh, G. Murugesan, M. J. Auxilius Jude, A. Harshini, S. Bhavataarani, and R. G. Krishnan, “Impacts of objective function on RPL-routing protocol: A survey,” in *2021 Sixth International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 251–255, 2021.

[35] B. Safaei, A. A. Mohammad Salehi, A. M. Hosseini Monazzah, and A. Ejlali, “Effects of RPL objective functions on the primitive characteristics of mobile and static iot infrastructures,” *Microprocessors and Microsystems*, vol. 69, pp. 79–91, 2019.

[36] H. Lamaazi and N. Benamar, “A comprehensive survey on enhancements and limitations of the RPL protocol: A focus on the objective function,” *Ad Hoc Networks*, vol. 96, p. 102001, 2020.

[37] D. Barthel, J. Vasseur, K. Pister, M. Kim, and N. Dejean, “Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks.” RFC 6551, Mar. 2012.

[38] P. Thubert, “Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL).” RFC 6552, Mar. 2012.

[39] O. Gnawali and P. Levis, “The Minimum Rank with Hysteresis Objective Function.” RFC 6719, Sept. 2012.

[40] H. Fotouhi, D. Moreira, and M. Alves, “mRPL: Boosting mobility in the internet of things,” *Ad Hoc Networks*, vol. 26, pp. 17–35, 2015.

[41] C. Cobârzan, J. Montavont, and T. Noël, “Analysis and performance evaluation of RPL under mobility,” in *2014 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, 2014.

[42] I. E. Korbi, M. Ben Brahim, C. Adjih, and L. A. Saidane, “Mobility enhanced RPL for wireless sensor networks,” in *2012 Third International Conference on The Network of the Future (NOF)*, pp. 1–8, 2012.

[43] P. Fabian, A. Rachedi, and C. Guéguen, “Programmable objective function for data transportation in the internet of vehicles,” *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 5, p. e3882, 2020.

[44] S. Hoghooghi and R. Javidan, “Proposing a new method for improving RPL to support mobility in the internet of things,” *IET Networks*, vol. 9, no. 2, pp. 48–55, 2020.

[45] V. Marques and J. Kniess, “Mobility aware RPL (maRPL): Providing mobility support for RPL protocol,” in *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, (Porto Alegre, RS, Brasil), pp. 211–223, SBC, 2019.

[46] F. Österlind, “A sensor network simulator for the contiki os,” 2006.

[47] A. Dunkels, B. Gronvall, and T. Voigt, “Contiki - a lightweight and flexible operating system for tiny networked sensors,” in *29th Annual IEEE International Conference on Local Computer Networks*, pp. 455–462, 2004.

[48] B. Safaei, A. Mohammadsalehi, K. T. Khoosani, S. Zarbaf, A. M. H. Monazzah, F. Samie, L. Bauer, J. Henkel, and A. Ejlali, “Impacts of mobility models on RPL-based mobile iot infrastructures: An evaluative comparison and survey,” *IEEE Access*, vol. 8, pp. 167779–167829, 2020.

[49] C. Y. Aung, B. C. Seet, M. Zhang, L. F. Xie, and P. H. J. Chong, “A review of group mobility models for mobile ad hoc networks,” *Wireless Personal Communications*, vol. 85, pp. 1317–1331, Dec 2015.

[50] M. U. Rahman, A. Alam, and S. Abbas, “Investigating the impacts of entity and group mobility models in manets,” in *2016 International Conference on Computing, Electronic and Electrical Engineering (ICE Cube)*, pp. 181–185, 2016.

[51] S. Manjula, A. C N, K. Shaila, V. K R, and L. Patnaik, “Performance of aodv routing protocol using group and entity mobility models in wireless sensor networks,” *Lecture Notes in Engineering and Computer Science*, vol. 2169, 03 2008.

[52] F. Bai, N. Sadagopan, and A. Helmy, “Important: a framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks,” in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, vol. 2, pp. 825–835 vol.2, 2003.

[53] A. Hanggoro and R. F. Sari, “Performance evaluation of the manhattan mobility model in vehicular ad-hoc networks for high mobility vehicle,” in *2013 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, pp. 31–36, 2013.

[54] S. Gowrishankar, S. Sarkar, and T. Basavaraju, “Simulation based performance comparison of community model, gfmm, rpgm, manhattan model and rwp-ss mobility models in manet,” in *2009 First International Conference on Networks Communications*, pp. 408–413, 2009.

[55] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn, “Bonnmotion: A mobility scenario generation and analysis tool,” in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, SIMUTools ’10, (Brussels, BEL), ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010.

[56] B. Salah Eddine, S. Omar, B. Meftah, M. Rebbah, and B. Cousin, “A reliable multipath routing protocol based on link quality and stability for manets in urban areas,” *Simulation Modelling Practice and Theory*, vol. 113, pp. 1–17, Id. : 102397, 08 2021.

[57] A. Elsts, “Tsch-sim: Scaling up simulations of tsch and 6tisch networks,” *Sensors*, vol. 20, no. 19, 2020.

[58] C. M. Ramya, M. Shanmugaraj, and R. Prabakaran, “Study on zigbee technology,” in *2011 3rd International Conference on Electronics Computer Technology*, vol. 6, pp. 297–301, 2011.

[59] R. Natarajan, P. Zand, and M. Nabi, “Analysis of coexistence between IEEE 802.15.4, ble and IEEE 802.11 in the 2.4 ghz ism band,” in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pp. 6025–6032, 2016.

[60] N. H. Hussein, C. T. Yaw, S. P. Koh, S. K. Tiong, and K. H. Chong, “A comprehensive survey on vehicular networking: Communications, applications, challenges, and upcoming research directions,” *IEEE Access*, vol. 10, pp. 86127–86180, 2022.

inclusão de apêndice

APÊNDICE A – CÓDIGO RSU

```
#include "contiki.h"
#include "net/routing/routing.h"
#include "net/netstack.h"
#include "net/ipv6/simple-udp.h"
#include "messages.h"
#include "sys/log.h"
#include "node-id.h"
#include "contiki-net.h"
#include "net/ipv6/uip-ds6.h"
#include "net/ipv6/uip-ds6-route.h"
#include "net/ipv6/uip-sr.h"
#include "net/ipv6/uip-debug.h"

#define LOG_MODULE "App"
#define LOG_LEVEL LOG_LEVEL_INFO

#define UDP_CLIENT_PORT 8765
#define UDP_SERVER_PORT 5678

uip_ipaddr_t own_ip;

static struct simple_udp_connection udp_conn;
static process_event_t accident_message_event;

PROCESS(send_accident_msg, "Send accident");
PROCESS(udp_server_process, "UDP server");


AUTOSTART_PROCESSES(&send_accident_msg,&udp_server_process);
PROCESS_THREAD(send_accident_msg, ev, data)
{
  PROCESS_BEGIN();
  accident_message_event = process_alloc_event();

  static uip_sr_node_t *node;

  while (1) {
    PROCESS_WAIT_EVENT_UNTIL(ev == accident_message_event);
    node = uip_sr_node_head();
    while(node != NULL) {
      static uip_ipaddr_t node_addr;
      NETSTACK_ROUTING.get_sr_node_ipaddr(&node_addr, node);
      if (uip_ipaddr_cmp(&node_addr, &own_ip)) {
        node = uip_sr_node_next(node);
        continue;
      }
      int node_id_rcv, msg_type, num_seq, x, y, z;
```

```
48
49      sscanf((const char *)data, "Node Id: %d num_seq: %d msg type: %d
            x: %d y: %d z: %d", &node_id_rcv, &num_seq, &msg_type, &x, &
           y, &z);
50      static char str[64];
51      snprintf(str, sizeof(str), "num_seq: %d msg type: %d", num_seq,
           ACCIDENT_REPORT_DISSEMINATION);
52      LOG_INFO("Sent packet ");
53      LOG_INFO_6ADDR(&own_ip);
54      LOG_INFO_(" '%.*s' to ", (int)strlen(str), (char *)str);
55      LOG_INFO_6ADDR(&node_addr);
56      LOG_INFO_("\n");
57      simple_udp_sendto(&udp_conn, str, strlen(str) + 1, &node_addr);
58      node = uip_sr_node_next(node);
59    }
60  }
61  PROCESS_END();
62 }
63
64 static void
65 udp_rx_callback(struct simple_udp_connection *c,
66        const uip_ipaddr_t *sender_addr,
67        uint16_t sender_port,
68        const uip_ipaddr_t *receiver_addr,
69        uint16_t receiver_port,
70        const uint8_t *data,
71        uint16_t datalen)
72 {
73  own_ip = *receiver_addr;
74  const uip_ipaddr_t receiver_addr_copy = *receiver_addr;
75  LOG_INFO("Received packet ");
76  LOG_INFO_6ADDR(&receiver_addr_copy);
77  LOG_INFO_(" '%.*s' from ", datalen, (char *) data);
78  LOG_INFO_6ADDR(sender_addr);
79  LOG_INFO_("\n");
80  int node_id_rcv, num_seq, msg_type, x, y, z;
81  sscanf((const char *)data, "Node Id: %d num_seq: %d msg type: %d x:
       %d y: %d z: %d",&node_id_rcv, &num_seq, &msg_type, &x, &y, &z);
82  switch(msg_type) {
83    case ACCIDENT_REPORT:
84      process_post(&send_accident_msg, accident_message_event, (void
           *)data);
85      send_accident_report_to_rsus(datalen, (char *) data);
86    break;
87    case ACCIDENT_REPORT_DISSEMINATION:
88      process_post(&send_accident_msg, accident_message_event, (void
           *)data);
89    break;
90  }
91
92 }
```

```
93
94  PROCESS_THREAD(udp_server_process, ev, data)
95  {
96    PROCESS_BEGIN();
97    own_ip = get_own_ip();
98    /* Set ipaddr with DODAG ID, so we get the prefix */
99    static uip_ipaddr_t ipaddr;
100   /* Initialize DAG root */
101   NETSTACK_ROUTING.root_start();
102   NETSTACK_ROUTING.get_root_ipaddr(&ipaddr);
103   LOG_INFO("Node ID %d has IP ", node_id);
104   LOG_INFO_6ADDR(&ipaddr);
105   LOG_INFO_("\n");
106   /* Initialize UDP connection */
107   simple_udp_register(&udp_conn, UDP_SERVER_PORT, NULL,
108                       UDP_CLIENT_PORT, udp_rx_callback);
109
110   print_own_ipv6();
111   PROCESS_END();
112 }
```

APÊNDICE B – CÓDIGO VEÍCULO

```
#include "contiki.h"
#include "net/routing/routing.h"
#include "net/netstack.h"
#include "net/ipv6/simple-udp.h"
#include <stdint.h>
#include <inttypes.h>
#include "messages.h"
#include "node-id.h"
#include "contiki-net.h"
#include "sys/node-id.h"

#include "sys/log.h"
#define LOG_MODULE "App"
#define LOG_LEVEL LOG_LEVEL_INFO


#define UDP_CLIENT_PORT 8765
#define UDP_SERVER_PORT 5678

#define SEND_INTERVAL     (1 * CLOCK_SECOND)
#define TIME_TO_CRASH     (20 * CLOCK_SECOND)

static struct simple_udp_connection udp_conn;
static int num_seq = 0;

PROCESS(udp_client_process, "UDP client");
AUTOSTART_PROCESSES(&udp_client_process);

static void
udp_rx_callback(struct simple_udp_connection *c,
       const uip_ipaddr_t *sender_addr,
       uint16_t sender_port,
       const uip_ipaddr_t *receiver_addr,
       uint16_t receiver_port,
       const uint8_t *data,
       uint16_t datalen)
{
  LOG_INFO("Received packet ");
  LOG_INFO_6ADDR(receiver_addr);
  LOG_INFO_(" '%.*s' from ", datalen, (char *) data);
  LOG_INFO_6ADDR(sender_addr);
  LOG_INFO_("\n");
}

PROCESS_THREAD(udp_client_process, ev, data)
{

```

```
48   PROCESS_BEGIN();
49
50   /* Initialize UDP connection */
51   simple_udp_register(&udp_conn, UDP_CLIENT_PORT, NULL,
52                       UDP_SERVER_PORT, udp_rx_callback);
53
54   print_own_ipv6();
55
56   #ifdef BROKEN_VEHICLE
57   uip_ipaddr_t own_ip = get_own_ip();
58   static struct etimer periodic_timer;
59   static struct etimer crash_timer;
60   static char str[64];
61
62   uip_ipaddr_t dest_ipaddr;
63   etimer_set(&crash_timer,TIME_TO_CRASH);
64   PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&crash_timer));
65
66   etimer_set(&periodic_timer, SEND_INTERVAL);
67   while(1) {
68     PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&periodic_timer));
69     if(NETSTACK_ROUTING.get_root_ipaddr(&dest_ipaddr)) {
70       num_seq++;
71       /* Send to DAG root */
72       snprintf(str, sizeof(str), "Node Id: %d num_seq: %d msg type: %d
             x: %d y: %d z: %d", node_id, num_seq, ACCIDENT_REPORT,
           x_location, y_location, z_location);
73       simple_udp_sendto(&udp_conn, str, strlen(str), &dest_ipaddr);
74       LOG_INFO("Sent packet ");
75       LOG_INFO_6ADDR(&own_ip);
76       LOG_INFO_(" '%.*s' to ", (int)strlen(str), (char *)str);
77       LOG_INFO_6ADDR(&dest_ipaddr);
78       LOG_INFO_("\n");
79       etimer_set(&periodic_timer, SEND_INTERVAL);
80     } else {
81       LOG_INFO("There is no root yet\n");
82     }
83   }
84   #endif
85
86
87   PROCESS_END();
88 }
```