



Universidade Federal da Paraíba
Centro de Energias Alternativas e Renováveis
Programa de Pós-Graduação em Engenharia Elétrica



Douglas de Farias Medeiros

Implementação e Análise de Protocolos de Roteamento para Redes *Mesh* sem Fio LoRa

João Pessoa

2021

Douglas de Farias Medeiros

Implementação e Análise de Protocolos de Roteamento para Redes *Mesh* sem Fio LoRa

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica - PPGEE, da Universidade Federal da Paraíba - UFPB, como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Cleonilson Protásio de Souza

Coorientador: Prof. Dr. Fabrício Braga Soares de Carvalho

João Pessoa

2021

Catálogo na publicação
Seção de Catalogação e Classificação

M488i Medeiros, Douglas de Farias.

Implementação e análise de protocolos de roteamento
para redes mesh sem fio LoRa / Douglas de Farias

Medeiros. - João Pessoa, 2021.

123 f. : il.

Orientação: Cleonilson Protásio de Souza.

Coorientação: Fabrício Braga Soares de Carvalho.

Dissertação (Mestrado) - UFPB/CEAR.

1. Redes - computadores - sensores sem fio. 2.
Roteamento - protocolos. 3. Cupcarbon - simulador. 4.
LoRa - comunicação sem fio. I. Souza, Cleonilson
Protásio de. II. Carvalho, Fabrício Braga Soares de.
III. Título.

UFPB/BC

CDU 004.7(043)

UNIVERSIDADE FEDERAL DA PARAÍBA – UFPB
CENTRO DE ENERGIAS ALTERNATIVAS E RENOVÁVEIS – CEAR
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA - PPGE

A Comissão Examinadora, abaixo assinada, aprova a Dissertação

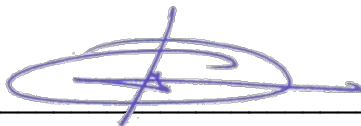
**IMPLEMENTAÇÃO E ANÁLISE DE PROTOCOLOS DE ROTEAMENTO PARA
REDES MESH SEM FIO LORA**

Elaborada por

DOUGLAS DE FARIAS MEDEIROS

como requisito parcial para obtenção do grau de
Mestre em Engenharia Elétrica.

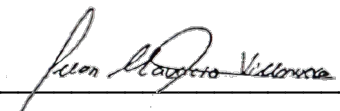
COMISSÃO EXAMINADORA



PROF. DR. CLEONILSON PROTASIO DE SOUZA
Orientador – UFPB



PROF. DR. FABRÍCIO BRAGA SOARES DE CARVALHO
Coorientador – UFPB



PROF. DR. JUAN MOISES MAURICIO VILLANUEVA
Examinador Interno – UFPB



PROF. DR. RUAN DELGADO GOMES
Examinador Externo – IFPB

Dedico este trabalho à minha família, que sempre foi minha base durante toda a vida, e também a todos os que sempre me apoiaram.

Agradecimentos

Primeiramente queria agradecer a Deus por me possibilitar desenvolver este trabalho durante uma época tão conturbada no mundo e sempre mantendo o foco.

Gostaria de agradecer aos meus orientadores por todo apoio concedido durante este período do mestrado, os quais me abriram diversas portas fundamentais nesta caminhada e me forneceram valiosos ensinamentos.

Agradecer também aos meus familiares e amigos, em especial a minha irmã Mariana e a minha namorada Jéssica.

Aos colegas de laboratório, que também sempre estiveram muito presentes em todas as etapas de construção deste trabalho.

De antemão também queria agradecer aos professores convidados para minha banca pela disponibilidade, sei que terão muito a contribuir.

Agradeço também à CAPES pelo fomento às minhas atividades de pesquisa.

Enfim, gostaria de agradecer a todos que contribuíram de alguma maneira para realização deste trabalho.

Resumo

Em um mundo cada vez mais conectado, torna-se evidente o crescimento do número de dispositivos que trocam dados entre si autonomamente. O aumento da conectividade desses dispositivos está diretamente relacionado aos conceitos de Internet das Coisas e Cidades Inteligentes, que têm ganhado muito destaque nos últimos anos. Sendo assim, à medida que cresce o número de dispositivos conectados e suas aplicações, a quantidade de dados compartilhados também aumentará de forma considerável. Logo, para que seja possível enviar dados de um dispositivo até outro e esses dados tenham que percorrer dispositivos intermediários, torna-se necessário definir por qual caminho os dados deverão ser encaminhados, caracterizando o processo de roteamento. Diante deste contexto, este trabalho tem como objetivo implementar e analisar diferentes protocolos de roteamento para aplicação em Cidades Inteligentes. Além disso, também foi desenvolvida e integrada ao simulador *Cupcarbon* uma ferramenta para cálculo do alcance dos nós utilizando o modelo de propagação de Egli. Em adição, este trabalho apresenta uma proposta de um algoritmo de ajuste de potência de transmissão visando a economia de energia dos nós da rede. Nas simulações, foi utilizada a tecnologia de comunicação sem fio LoRa (*Long Range*) e considerado um cenário de simulação composto por uma topologia de rede em malha distribuída por avenidas da cidade de João Pessoa-PB. Os critérios de avaliação utilizados foram a razão de entrega de pacotes, atraso médio fim-a-fim, *jitter* médio e a taxa de transferência. Também foram obtidos os perfis de consumo de cada nó sensor para cada protocolo. Os resultados mostraram que o protocolo DSR foi a opção mais adequada dentre as implementadas para ser utilizado em conjunto com o algoritmo de ajuste de potência de transmissão proposto neste trabalho, o qual proporcionou uma economia de energia de 11,32% em relação ao DSR original. Por outro lado, o protocolo AODV obteve melhor desempenho geral nas métricas e apresentou o segundo maior consumo de energia. Por fim, o protocolo DVR foi o que mais consumiu energia e obteve melhor desempenho no quesito latência, mas apresentou alta perda de pacotes. Ao final, foram realizados testes experimentais utilizando o protocolo DSR e uma aplicação no servidor *The Things Network* (TTN).

Palavras-chave: Redes de Sensores sem Fio, Protocolos de Roteamento, Simulador *Cupcarbon*, LoRa.

Abstract

In a world getting more and more connections, the growth of the number of devices sharing autonomously data with each other has been evident. The connectivity of all these devices is directly related to the concepts of Internet of Things and Smart Cities, which has gaining much attention currently. Therefore, as the number of connected devices and their applications grows, the amount of data shared by these devices also increases considerably. Thus, in order to be able to send data from one device to another passing through intermediate devices, it is necessary to define the path to the data to be sent, characterizing the concept of routing. In this context, this work aims to implement and analyze different routing protocols for application in Smart Cities. In addition, a tool for calculating the distance range of nodes using the Egli propagation model was also developed and integrated on Cupcarbon simulator. Furthermore, this work also showed a proposal for a transmission power adjustment algorithm with the objective of saving energy at the network nodes. The LoRa wireless communication technology and a scenario of simulation composed by a mesh network topology distributed along avenues in the city of João Pessoa - PB were considered. The evaluation criteria were the packet delivery rate, the average end-to-end delay, the average jitter and throughput. The energy consumption profiles of each network node were also obtained according to each protocol. The results showed that the DSR protocol was the most appropriate option among those implemented to be used with the transmission power adjustment algorithm proposed in this work. This algorithm provided an energy saving of 11.32% compared to the original DSR. On the other hand, the AODV protocol achieved better general performance and presented the second highest energy consumption. Finally, the DVR protocol was the one that consumed the most energy and performed better in terms of latency, but presented high packet loss. In the end, experimental tests were performed using the DSR protocol and an application on The Things Network (TTN) server.

Keywords: Wireless Sensor Networks, Routing Protocols, Cupcarbon Simulator, LoRa.

Lista de siglas

- ALOHA** *Additive Links On-Line Hawaii Area*
- AODV** *Ad Hoc On-Demand Distance Vector*
- Chirp** *Compressed High Intensity Radar Pulse*
- CSS** *Chirp Spread Spectrum*
- DSR** *Dynamic Source Routing*
- DVR** *Distance Vector Routing*
- E2ED** *End-to-end Delay*
- IoT** *Internet of Things*
- JIT** *Jitter*
- LoRa** *Long Range*
- LoRaWAN** *LoRa Wide Area Network*
- LPWAN** *Low Power Wide Area Network*
- MAC** *Media Access Control*
- NB-IoT** *Narrowband-IoT*
- NS** *Network Simulator*
- PDR** *Packet Delivery Ratio*
- RSSF** *Redes de Sensores sem Fio*
- RSSFL** *Redes de Sensores sem Fio Lineares*
- RSSI** *Received Signal Strength Indication*
- THR** *Throughput*
- Wi-SUN** *Wireless Smart Ubiquitous Networks*

Sumário

Lista de Figuras	xii
Lista de Tabelas	xiv
Lista de Quadros	xv
1 Introdução	16
1.1 Motivação	17
1.2 Trabalhos Relacionados	18
1.3 Objetivos	20
1.3.1 Objetivo Geral	20
1.3.2 Objetivos Específicos	20
1.4 Organização do Texto	21
2 Redes de Sensores Sem Fio (RSSF)	22
2.1 Definições Gerais	22
2.1.1 Composição de uma RSSF	23
2.1.2 Topologias de Rede	24
2.2 LPWAN	25
2.2.1 LoRa	26
2.2.2 Sigfox	29
2.2.3 NB-IoT	30
2.2.4 Wi-SUN	30
3 Simuladores de Rede	32
3.1 <i>Network Simulator</i> (NS-2)	32
3.2 <i>Objective Modular Network Testbed</i> (OMNeT++)	33
3.3 <i>Riverbed Modeler</i>	35
3.4 <i>Cupcarbon</i>	36
4 Modelos de Perda de Propagação	41
4.1 Modelo de Espaço Livre	42

4.2	Modelo de Okumura	42
4.3	Modelo de Okumura-Hata	44
4.4	Modelo COST 231	45
4.5	Modelo de Egli	46
5	Protocolos de Roteamento para RSSF	47
5.1	Visão Geral sobre Roteamento	47
5.1.1	Algoritmos Vetor-Distância	48
5.1.2	Algoritmos de Estado de <i>link</i>	48
5.2	Classificação dos Protocolos para Redes <i>ad hoc</i>	49
5.2.1	Pró-ativos	50
5.2.1.1	<i>Distance Vector Routing Protocol</i> (DVR)	51
5.2.1.2	<i>Destination Sequenced Distance Vector</i> (DSDV)	53
5.2.2	Reativos	55
5.2.2.1	<i>Ad Hoc On-Demand Distance Vector</i> (AODV)	55
5.2.2.2	<i>Dynamic Source Routing</i> (DSR)	59
5.2.3	Híbridos	60
6	Metodologia	61
6.1	Descrição e Estrutura de Implementação dos Protocolos	61
6.1.1	Organização dos Códigos	62
6.1.2	Implementação dos Protocolos	63
6.1.3	Implementação da Mobilidade	67
6.2	Desenvolvimento de Ferramenta para Cálculo do Raio de Alcance	68
6.3	Cenário de Simulação e Topologia de Rede	70
6.4	Métricas de Avaliação de Desempenho	74
6.4.1	Razão de Entrega de Pacotes	74
6.4.2	Atraso Médio Fim-a-Fim	75
6.4.3	<i>Jitter</i> Médio	75
6.4.4	Taxa de Transferência	76
6.4.5	Modelos de Consumo de Energia	76
6.4.5.1	Modelo Clássico	77

6.4.5.2	Modelo de Heinzelman	78
6.5	Algoritmo de Ajuste de Potência de Transmissão	78
6.5.1	Cálculo do RSSI no <i>Cupcarbon</i>	79
6.5.2	Equação de Potência de Transmissão versus RSSI	80
6.5.3	Fluxograma	82
6.6	Descrição da Implementação Experimental	84
7	Resultados e Discussões	88
7.1	Resultados de Simulação	89
7.1.1	Economia de Energia do Ajuste de Potência	89
7.1.2	Razão de Entrega de Pacotes	90
7.1.3	Atraso Médio Fim-a-Fim	91
7.1.4	<i>Jitter</i> Médio	93
7.1.5	Taxa de Transferência (<i>Throughput</i>)	93
7.1.6	Consumo Médio de Energia	94
7.2	Resultados Experimentais	97
7.2.1	Etapas de Roteamento	97
7.2.2	Teste Completo de Roteamento	101
8	Conclusões	105
8.1	Trabalhos Futuros	106
8.2	Publicações	107
	Referências	109
	Apêndices	117

Lista de Figuras

1	Exemplo de uma rede de sensores sem fio (RSSF).	23
2	Tipos de topologias para RSSF <i>Zigbee</i>	24
3	Tecnologias de comunicação sem fio.	25
4	Pilha do protocolo LoRa/LoRaWAN.	27
5	Topologia de uma rede LoRaWAN.	28
6	Exemplo de pacote de dados modulado em LoRa.	29
7	Arquitetura do <i>software</i> de simulação de redes NS-2.	33
8	Interface gráfica do simulador OMNeT++.	34
9	Interface gráfica do simulador <i>Riverbed Modeler</i>	35
10	Fluxo de funcionamento do simulador <i>Riverbed Modeler</i>	36
11	Interface gráfica do simulador <i>Cupcarbon</i>	37
12	Arquitetura do simulador <i>Cupcarbon</i>	38
13	Exemplo de exibição de mapa em três dimensões no <i>Cupcarbon</i>	39
14	Curvas do modelo de Okumura.	44
15	Classificação dos protocolos de roteamento para redes <i>ad hoc</i> sem fio. . .	49
16	Topologia de exemplo do protocolo DVR.	52
17	Exemplo de funcionamento do protocolo AODV.	57
18	Exemplo de funcionamento do protocolo DSR. (a) RREQ (b) RREP. . . .	59
19	Exemplo de notificação de indisponibilidade de rota do protocolo DSR. . .	60
20	Diagrama com a estrutura de implementação dos protocolos de roteamento.	61
21	Diagrama de organização dos códigos.	63
22	Fluxograma de implementação do roteamento do protocolo DVR.	64
23	Fluxograma de implementação do roteamento do protocolo AODV.	65
24	Fluxograma de implementação do roteamento do protocolo DSR.	66
25	Fluxograma de mobilidade.	67
26	Integração da ferramenta auxiliar com a interface gráfica do <i>Cupcarbon</i> . .	69
27	Interface gráfica da ferramenta auxiliar para cálculo do raio de alcance. . .	70
28	Topologia de rede utilizada nas simulações.	71
29	Trajetórias utilizadas no cenário de simulação.	73

30	Simulação realizada para obtenção da expressão de potência de transmissão versus RSSI.	80
31	Comparação dos polinômios de aproximação dos dados experimentais de potência de transmissão e RSSI.	81
32	Fluxogramas do ajuste de P_{tx} . (a) Identificação do nível de potência de transmissão de um nó de destino. (b) Ajuste de potência antes de transmitir dados.	83
33	Instalação do <i>gateway</i> industrial no Campus I da Universidade Federal da Paraíba.	84
34	Topologia da montagem experimental.	85
35	Esquema de ligação entre a placa ESP32 e o módulo de comunicação SX1276.	85
36	Fluxograma do código de interface do nó concentrador.	86
37	Exemplo de apresentação do cálculo das métricas no <i>console</i> do <i>Cupcarbon</i>	88
38	Margem de segurança do ajuste de potência versus economia de energia.	90
39	Resultados de simulação para a métrica PDR.	91
40	Resultados de simulação para a métrica E2ED.	92
41	Resultados de simulação para a métrica JIT.	93
42	Resultados de simulação para a métrica THR.	94
43	Consumo médio de energia de cada nó por protocolo.	95
44	Consumo médio total por protocolo.	97
45	Teste experimental das etapas de roteamento com 1 nó.	98
46	Teste experimental das etapas de roteamento com 2 nós.	98
47	Teste experimental das etapas de roteamento com 3 nós.	99
48	Confirmação das etapas de roteamento.	99
49	Teste de indisponibilidade de rota.	100
50	Teste de indisponibilidade de rota.	101
51	<i>Console</i> dos nós para o teste completo de roteamento.	103
52	Confirmação de recebimento dos dados na aplicação do TTN.	103
53	Detalhes de uma das mensagens recebidas no TTN.	104

Lista de Tabelas

1	Alcance esperado do LoRa de acordo com o tipo de ambiente.	26
2	Parâmetros gerais utilizados nas simulações.	72
3	Consumo de corrente do módulo SX1276 de acordo com o modo de operação.	77
4	RMSE para as curvas de aproximação de potência de transmissão versus RSSI.	82
5	Resultados de simulação do protocolo DVR para métrica PDR.	117
6	Resultados de simulação do protocolo AODV para métrica PDR.	117
7	Resultados de simulação do protocolo DSR para métrica PDR.	118
8	Resultados de simulação do protocolo DSR modificado para métrica PDR.	118
9	Resultados de simulação do protocolo DVR para métrica E2ED.	119
10	Resultados de simulação do protocolo AODV para métrica E2ED.	119
11	Resultados de simulação do protocolo DSR para métrica E2ED.	120
12	Resultados de simulação do protocolo DSR modificado para métrica E2ED.	120
13	Resultados de simulação do protocolo DVR para métrica JIT.	121
14	Resultados de simulação do protocolo AODV para métrica JIT.	121
15	Resultados de simulação do protocolo DSR para métrica JIT.	122
16	Resultados de simulação do protocolo DSR modificado para métrica JIT.	122
17	Resultados de simulação do protocolo DVR para métrica THR.	123
18	Resultados de simulação do protocolo AODV para métrica THR.	123
19	Resultados de simulação do protocolo DSR para métrica THR.	124
20	Resultados de simulação do protocolo DSR modificado para métrica THR.	124

Lista de Quadros

1	Tabela de roteamento inicial do nó A (Exemplo do protocolo DVR) . . .	51
2	Algoritmo DVR.	52
3	Tabela de roteamento final do nó A (Exemplo do protocolo DVR)	53
4	Algoritmo AODV.	58
5	Parâmetros para estimação do consumo de energia.	76

1 Introdução

Em um mundo cada vez mais conectado, torna-se evidente o crescimento do número de dispositivos trocando dados entre si de forma autônoma. Segundo projeção do IoT Analytics (LUETH, 2018), o número de dispositivos conectados em todo o mundo atingirá cerca de 25 bilhões até o final do ano de 2022, enquanto que o portal Statista indica que esse número pode chegar a 30 bilhões (Statista Analytics, 2016).

O aumento da conectividade de todos esses dispositivos está diretamente relacionado ao conceito de Internet das Coisas (IoT, do inglês *Internet of Things*), termo criado há mais de 20 anos (ASTHON, 2010). A perspectiva de IoT visa o estabelecimento de uma rede de dispositivos físicos conectados à Internet e equipados com microcontroladores e transceptores, tornando-os capazes de se comunicar autonomamente uns com os outros e também com usuários (ZANELLA *et al.*, 2014). Logo, o termo IoT tem ganhado bastante destaque atualmente, especialmente devido ao crescimento de aplicações em áreas como, por exemplo, indústria, meios de transporte e no contexto das cidades inteligentes (SCULLY, 2020).

Uma rede de sensores sem fio (RSSF), por sua vez, é constituída por um conjunto de nós sensores distribuídos em uma determinada área e que foi projetada para executar tarefas de controle e monitoramento, possibilitando o compartilhamento dos dados coletados entre si (BHANU; JASMINE; MAHADEVASWAMY, 2020). As RSSF são diretamente aplicadas no contexto de IoT.

Portanto, ao passo que o número de dispositivos conectados se expande, a quantidade de dados compartilhados também aumenta substancialmente em função desse crescimento, deixando clara a importância da discussão sobre como esta demanda de transmissão de dados deverá ser executada. Assim, para que seja possível enviar um pacote de dados até outro dispositivo fora de alcance, mas conectado em rede, é necessário que a informação seja encaminhada por diversos dispositivos intermediários, caracterizando o processo de roteamento de dados.

Diante desse cenário, os protocolos de roteamento assumem o papel de tomada de decisão em relação às rotas pelas quais os dados deverão ser encaminhados levando em consideração diferentes critérios, tais como o nível de carga da bateria de cada nó da rede ou o número de saltos, por exemplo (FARIAS, 2008).

1.1 Motivação

O crescimento urbano foi potencializado nas últimas décadas não só por fatores econômicos, mas também políticos, sociais e de saúde. Entretanto, este crescimento traz consigo problemas tais como dificuldade na gestão de resíduos, problemas de mobilidade, escassez de recursos naturais, poluição sonora e do ar, dentre outros. (GOMES *et al.*, 2020). A poluição do ar, por exemplo, é um problema ambiental que pode causar diversas doenças nos seres humanos e danos ao meio ambiente e aos animais, sendo os meios de transporte e as indústrias as principais fontes de poluentes na atmosfera (VILLARIM *et al.*, 2020).

Dentro dessa perspectiva, as Cidades Inteligentes (ou *Smart Cities*) surgem como alternativa para viabilizar aplicações que visam solucionar diversos problemas associadas aos centros urbanos, como por exemplo, monitoramento da poluição atmosférica. Desta maneira, as Cidades Inteligentes constituem cenários urbanos que utilizam Tecnologia da Informação e Comunicação (TIC) para melhorar a infraestrutura e a qualidade de vida dos cidadãos (AGUIAR, 2018). Assim, alinhado aos conceitos de IoT e RSSF, as Cidades Inteligentes proporcionam meios de realizar aquisição, transmissão e processamento de dados e possibilitam gerar meios eficazes para o enfrentamento dos desafios do meio urbano (ZANELLA *et al.*, 2014).

A implementação de uma RSSF é um processo que compreende diversas etapas, tendo como exemplo os tipos de sensores que serão utilizados em função da aplicação, o número de sensores e a topologia de rede, a tecnologia de comunicação, assim como o protocolo que será utilizado para realizar o encaminhamento de dados entre os dispositivos da rede.

Os protocolos de roteamento para redes *ad hoc* sem fio, por sua vez, são classificados quanto à sua natureza e propriedades, levando em consideração diferentes fatores, tais como quando e como as rotas são calculadas (SRIVASTAVA *et al.*, 2014; DEBBARMA; SEN; ROY, 2012). Portanto, a escolha para utilização de um determinado protocolo é muito importante em um projeto de RSSF e pode ser auxiliada por meio de uma análise do seu desempenho de acordo com diferentes cenários e aplicações, levando em consideração suas principais características.

Nesse sentido, as simulações computacionais mostram-se como importantes ferramentas de projeto de RSSF, possibilitando avaliar o comportamento dos protocolos de rotea-

mento em cenários de alta densidade de nós sensores e de aplicação real, por exemplo, reduzindo custos e economizando tempo de implementação (AGUIAR, 2018). Logo, as simulações podem auxiliar na escolha de um determinado protocolo, além de permitir a validação de novos protocolos em cenários submetidos a situações desfavoráveis, a exemplo da ocorrência de falhas nos dispositivos.

Diante desse contexto, o objetivo desta Dissertação é implementar e avaliar o desempenho de protocolos de roteamento para o estabelecimento de redes LoRa em malha (*mesh*), visando aplicações de RSSF como, por exemplo, o monitoramento de variáveis associadas aos gases poluentes na atmosfera em centros urbanos. Para esta finalidade, será utilizado o simulador de redes *Cupcarbon*, o qual foi desenvolvido no projeto de pesquisa *Persepteur* especificamente para o cenário de Cidades Inteligentes e IoT. Em adição, também foram obtidos resultados experimentais dos protocolos de roteamento implementados na cidade de João Pessoa-PB.

1.2 Trabalhos Relacionados

Em Lee e Ke (2018), foi proposta uma aplicação de IoT utilizando uma rede de sensores sem fio distribuída em uma grande área geográfica, na qual os nós sensores comunicam-se por meio da tecnologia LoRa. Adicionalmente, os autores realizaram uma análise do desempenho da comunicação variando parâmetros relacionados à camada física LoRa, tais como largura de banda (*Bandwidth* - BW) e fator de espalhamento (*Spreading Factor* - SF).

Ahmad *et al.* (2019) desenvolveram uma aplicação de monitoramento remoto inteligente, no segmento de agricultura de precisão, que foi obtida pela combinação entre os conceitos de IoT e RSSF. Neste trabalho, os nós sensores foram distribuídos ao longo do campo e coletam dados sobre temperatura, umidade do solo, qualidade do ar, entre outras variáveis associadas à cultura de produtos agrícolas. Os dados coletados neste sistema foram transmitidos para uma estação base, utilizando uma rede de sensores baseada no protocolo Zigbee e encaminhados para uma nuvem computacional. Desta maneira, foi proporcionado aos agricultores uma plataforma visual na Internet com os dados obtidos.

Uma outra aplicação para RSSF foi apresentada por Aguiar (2018), em que é proposta uma rede de monitoramento de variáveis ambientais, tais como poluição atmosférica,

utilizando uma RSSF em linha (RSSFL). Nesta aplicação, foram realizadas simulações no *software* OMNet++/Castalia visando modelar situações que afetem a transmissão do sinal e em diferentes distâncias entre nós sensores de uma RSSFL, analisando a transmissão de acordo com métricas como taxa de erro de pacote (TEP) e indicador de intensidade de sinal recebido (RSSI).

Al-Dhief *et al.* (2018) realizaram experimentos de simulação no *software* NS-2 com o propósito de comparar o desempenho dos protocolos de roteamento DSR, AODV e DSDV. Para realizar a comparação entre os protocolos, foram obtidas as métricas de taxa de entrega de pacotes, taxa de transferência média e latência média de ponta a ponta em relação à variação do número de nós da rede. Após realizar as simulações, os autores chegaram à conclusão de que, dentro do contexto do trabalho, o protocolo DSDV foi o que apresentou melhor desempenho em termos de entrega de pacotes.

Trabalhos semelhantes ao de Al-Dhief *et al.* (2018) podem ser encontrados em alguns outros (MATRE; KARANDIKAR, 2016), (RAHMAN; HASAN; ISLAM, 2012), (MEDEIROS *et al.*, 2020) e (ROSHAN; MISHRA; MEHER, 2019). Nesses trabalhos, os autores também comparam o desempenho de diferentes protocolos de roteamento utilizando ferramentas de simulação e métricas tais como taxa de entrega de pacotes, latência média, *jitter* médio e taxa de transferência. São realizadas simulações sob diversos cenários, como, por exemplo, diferentes densidades de nós e também variação no tempo de pausa em relação à mobilidade.

Por fim, alguns trabalhos abordando protocolos de roteamento e redes LoRa podem ser encontrados em Lundell *et al.* (2018), Almeida *et al.* (2020) e Macaraeg, Hilario e Ambatali (2020). Em Lundell *et al.* (2018) os autores trazem uma proposta de protocolo de roteamento baseado no protocolo AODV para ser utilizado em redes LoRa em malha, enquanto que Almeida *et al.* (2020) propõem o desenvolvimento de uma rede híbrida composta por uma topologia em malha LoRa em conjunto com uma rede que utiliza o protocolo LoRaWAN. Macaraeg, Hilario e Ambatali (2020) desenvolveram um sistema de comunicação de emergências utilizando uma rede LoRa em malha; dessa forma, os autores implementaram uma versão modificada do protocolo AODV e avaliaram a viabilidade do sistema de acordo com a métrica de taxa de entrega de pacotes.

No entanto, o diferencial deste trabalho de Dissertação está no fato de ter realizado

implementações em um simulador de redes focado em aplicações de IoT e Cidades Inteligentes, assim como ter analisado o desempenho de diferentes protocolos de roteamento de acordo com um cenário de simulação de aplicação real, utilizando métricas quantitativas para fins de comparação. Adicionalmente, este trabalho também trouxe uma análise do consumo de energia dos nós da rede para todos os protocolos considerados, assim como uma implementação experimental de um desses protocolos.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral deste trabalho de Dissertação de Mestrado consiste na implementação e análise de diferentes protocolos de roteamento para redes sem fio em malha utilizando a tecnologia de comunicação sem fio LoRa, visando aplicações de RSSF no contexto de Cidades Inteligentes.

1.3.2 Objetivos Específicos

Dentre os objetivos específicos desta proposta, pode-se destacar:

- Implementar e validar os protocolos de roteamento no simulador de redes *Cupcarbon* considerando um cenário de simulação urbano, voltado para aplicações de RSSF tais como o monitoramento de gases poluentes na atmosfera;
- Desenvolver uma ferramenta dentro do simulador *Cupcarbon (plugin)* para cálculo do raio de alcance entre os dispositivos de acordo com o modelo de propagação de Egli e que leve em consideração os parâmetros da modulação LoRa;
- Calcular métricas quantitativas para comparação e análise dos resultados de desempenho dos protocolos considerados;
- Definir o protocolo mais adequado em relação ao desempenho geral, dentre os protocolos avaliados neste trabalho, para implementação de um algoritmo de ajuste de potência de transmissão visando a economia de energia da rede;
- Comparar os resultados de consumo entre o protocolo de roteamento original e o modificado com ajuste de potência proposto neste trabalho; e

- Avaliar experimentalmente um dos protocolos de roteamento na cidade de João Pessoa.

1.4 Organização do Texto

Esta Dissertação está organizada em oito capítulos, de acordo com a seguinte estrutura: no Capítulo 2 é apresentado o conceito de redes de sensores sem fio e suas principais definições. Além disso, neste capítulo também são destacadas as principais tecnologias que compõem as redes LPWAN (*Low Power Wide Area Networks*). No Capítulo 3 é realizada uma comparação entre as ferramentas de simulação de redes mais utilizadas. No Capítulo 4 são descritos alguns modelos de perda de propagação. Em seguida, no Capítulo 5 é mostrada uma visão geral sobre roteamento, apresentando alguns dos principais algoritmos de roteamento e também a classificação dos protocolos para redes *ad hoc* sem fio. Adicionalmente, no Capítulo 5 são descritos alguns protocolos que serão utilizados na etapa de simulação deste trabalho. Após apresentar os conceitos necessários para a compreensão das atividades desenvolvidas, é detalhada no Capítulo 6 a estrutura de implementação utilizada, a organização dos códigos e a topologia de rede adotada. No Capítulo 6 também são descritos o algoritmo de ajuste de potência proposto nesta Dissertação, assim como a implementação experimental que foi realizada. Os cenários de simulação também são descritos nesse capítulo, finalizando com uma explicação sobre as métricas quantitativas utilizadas para avaliação de desempenho dos protocolos implementados. No Capítulo 7 são apresentados e discutidos os resultados de simulação para os cenários considerados, assim como os resultados experimentais. Por fim, as conclusões deste trabalho e os trabalhos futuros são descritos no Capítulo 8.

2 Redes de Sensores Sem Fio (RSSF)

Neste capítulo são descritos o conceito de RSSF e algumas definições relacionadas a este tema. Também são apresentados alguns elementos geralmente utilizados nestas redes e suas principais topologias. O capítulo é finalizado apresentando as redes LPWAN (*Low Power Wide Area Network*) e as tecnologias de comunicação mais utilizadas nesse tipo de rede.

2.1 Definições Gerais

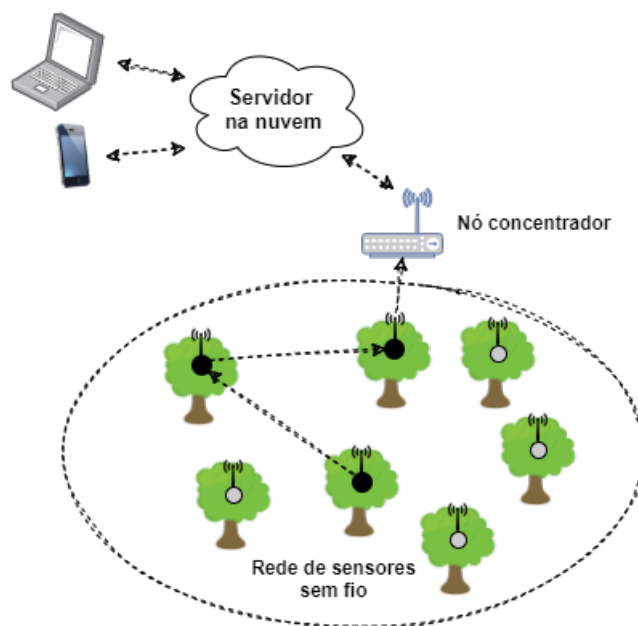
Os crescentes avanços das tecnologias de comunicação sem fio e da eletrônica, especialmente no que diz respeito ao consumo de energia e alcance de transmissão, favoreceram consideravelmente as aplicações das RSSF. Assim, dentro de um contexto em que dispositivos podem trocar dados entre si, as RSSF constituem um tipo especial de redes *ad hoc* e são compostas por um conjunto de dispositivos com diferentes funções, tais como *gateways*, roteadores e nós sensores. Estes dispositivos podem ser distribuídos de maneira aleatória em uma determinada área e são capazes de monitorar fenômenos e grandezas físicas (CORKE *et al.*, 2010; AGUIAR, 2018). Também é possível realizar tarefas de controle por meio dos nós sensores, quando dispositivos atuadores possibilitam controlar parâmetros de um ambiente (RUIZ *et al.*, 2004). Os dados coletados pelos nós sensores podem ser encaminhados para um nó concentrador ou *gateway*.

Muitas aplicações podem ser viabilizadas com a utilização das RSSF, principalmente devido ao baixo custo e à facilidade de instalação dos dispositivos, o que possibilita aplicações em áreas de difícil acesso, favorecendo a aquisição remota de dados (AKYILDIZ *et al.*, 2002). As RSSFs podem ser aplicadas no monitoramento de poluição, medição de condições meteorológicas (temperatura, velocidade do vento, radiação solar, precipitação atmosférica, etc.), incêndio florestal, atividade sísmica, etc. (CORKE *et al.*, 2010). Além disso, aplicações de monitoramento em larga escala também podem ser desenvolvidas com o uso de RSSF (AGUIAR, 2018).

2.1.1 Composição de uma RSSF

Uma estrutura típica de aplicação com RSSF é apresentada na Figura 1. Nesta imagem podem ser observados os principais componentes de uma RSSF, tais como os nós sensores, os quais estão representados pelos dispositivos acoplados em árvores neste exemplo.

Figura 1: Exemplo de uma rede de sensores sem fio (RSSF).



Fonte: Adaptado de Medeiros *et al.* (2020).

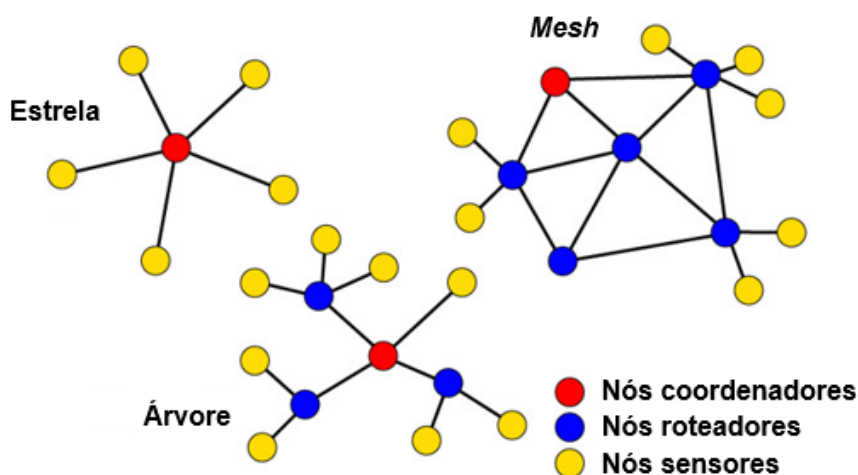
Uma RSSF é constituída essencialmente pelos seguintes elementos (AGUIAR, 2018):

- **Nó sensor:** este dispositivo é responsável por realizar a aquisição de dados nas aplicações de monitoramento e transmitir a informação por meio de nós intermediários para que chegue ao concentrador, que é o nó de destino;
- **Nó roteador:** possui a funcionalidade de receber os dados obtidos por outros nós da rede e encaminhá-los para outros roteadores ou para o nó concentrador;
- **Nó concentrador:** este dispositivo possui como função receber os dados dos nós sensores ou dos roteadores da rede sem fio e direcioná-los para uma estação central, como, por exemplo, um servidor em nuvem computacional, para que o usuário final possa acessá-los.

2.1.2 Topologias de Rede

As topologias de rede dizem respeito à maneira como os dispositivos são interligados, influenciando na maneira como a informação é encaminhada ao nó concentrador. Existem diversas topologias que podem ser utilizadas nas redes de sensores sem fio, sendo apresentadas na Figura 2 algumas topologias comuns para redes Zigbee, por exemplo.

Figura 2: Tipos de topologias para RSSF Zigbee.



Fonte: Adaptado de Elprocus (2020).

No exemplo da Figura 2, em que é considerado o protocolo Zigbee, é possível observar os principais tipos de topologias que podem ser utilizadas para implementação das RSSF. Na topologia em estrela, por exemplo, a rede é composta por um nó coordenador, responsável pelo gerenciamento dos dispositivos da rede e por diversos nós sensores, sendo o nó coordenador o único destino possível para encaminhamento dos dados (Elprocus, 2020).

Nas topologias em malha (*mesh*) e árvore estão presentes, além dos nós sensores e coordenadores, os nós roteadores, os quais podem encaminhar as informações para nós adjacentes. Uma característica da topologia em malha é a possibilidade de enviar dados para um mesmo destino por múltiplos caminhos. Dessa forma, caso ocorra algum tipo de falha em um nó, os dados podem ser encaminhados por meio de outros dispositivos.

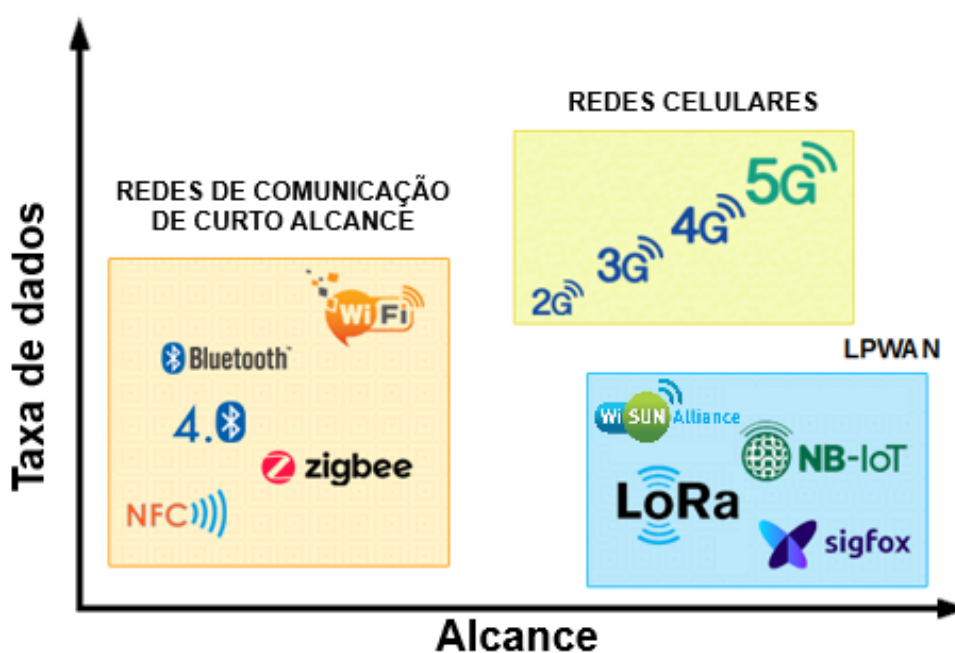
2.2 LPWAN

Existem diversas tecnologias de comunicação sem fio que possibilitam a implementação de uma RSSF, tal como *Zigbee*, exemplificado no tópico anterior. Assim, pode-se classificar algumas das tecnologias de comunicação sem fio mais utilizadas em aplicações de IoT e RSSF como:

- LPWAN (*Low Power Wide Area Network*);
- Redes celulares 3G/4G/5G;
- ZigBee.

Nesse contexto, as LPWANs têm ganhado destaque por apresentar características interessantes para estabelecimento de RSSF e em aplicações para IoT, tais como baixo consumo de energia e transmissões em longas distâncias. Na Figura 3 estão elencadas as principais redes de comunicação sem fio em uma perspectiva de alcance *versus* taxa de dados.

Figura 3: Tecnologias de comunicação sem fio.



Fonte: Adaptado de Kumar (2019).

As redes LPWANs foram desenvolvidas com o intuito de enviar pacotes de dados em longas distâncias e também com baixo consumo de energia, sendo esta última característica uma das suas principais vantagens em relação aos demais tipos de redes.

Como pode ser visto na Figura 3, alguns exemplos de tecnologias LPWAN são:

- *Narrowband IoT* (NB-IoT);
- Sigfox;
- LoRa;
- Wi-SUN.

2.2.1 LoRa

O LoRa (*Long Range*) é uma tecnologia de comunicação sem fio patenteada pela *Semtech Corporation* e foi criada com o intuito de ser utilizada em dispositivos com limitações de energia (como por exemplo, alimentados por baterias) visando longa duração e que necessitam de maior alcance de transmissão (AUGUSTIN *et al.*, 2016). Entretanto, o alcance entre o transmissor e o receptor depende, dentre outros fatores, do tipo de ambiente e obstáculos em que os equipamentos serão utilizados. Na Tabela 1 é apresentado o alcance esperado do LoRa para alguns tipos de ambientes.

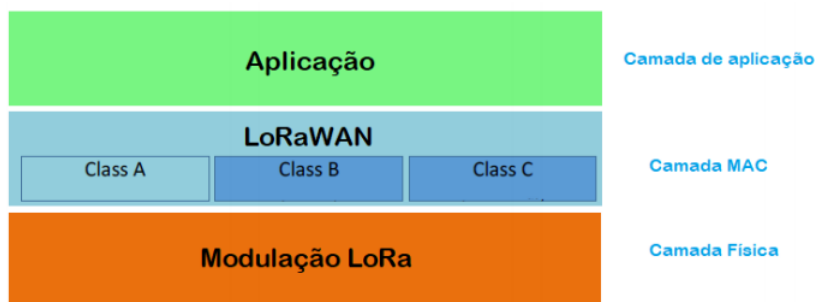
Tabela 1: Alcance esperado do LoRa de acordo com o tipo de ambiente.

Tipo de ambiente	Alcance médio (km)
Áreas urbanas	2 - 5
Áreas rurais	5 - 15
Linha de visada direta	>15

Fonte: Semtech (2020).

A pilha do protocolo da tecnologia LoRa/LoRaWAN está ilustrada na Figura 4 e pode ser situada em duas camadas do modelo OSI (*Open Systems Interconnection*). A modulação LoRa está na camada física, enquanto que o protocolo LoRaWAN encontra-se na camada MAC (do inglês, *Media Access Control*).

Figura 4: Pilha do protocolo LoRa/LoRaWAN.



Fonte: Adaptado de Semtech (2020).

Camada MAC

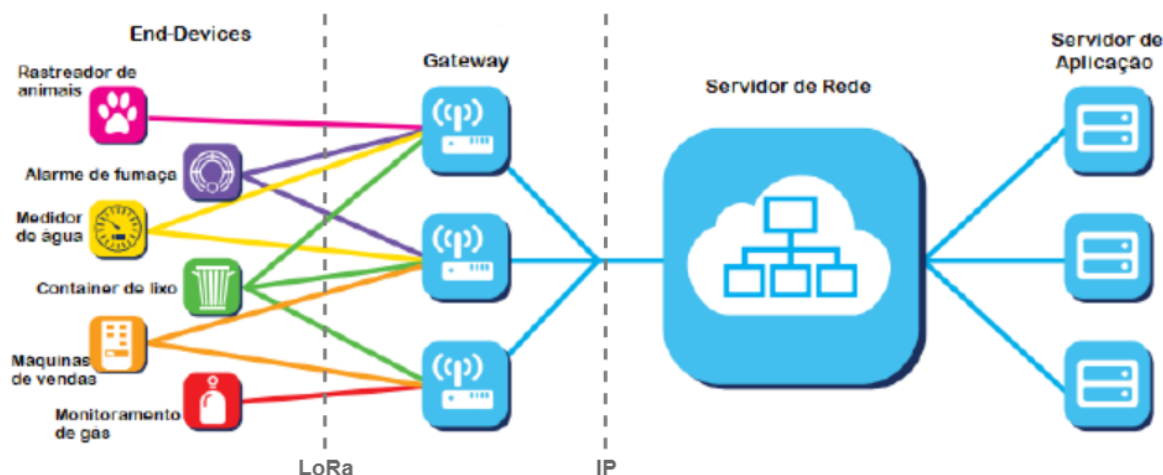
De acordo com LoRa Alliance (2021), LoRaWAN é a especificação de um protocolo de rede situado na camada MAC que foi desenvolvido para ser utilizado em conjunto com a modulação LoRa levando em consideração requisitos de RSSF e IoT (tais como comunicação bidirecional, segurança, mobilidade e baixo consumo de energia). O LoRaWAN proporciona um mecanismo de controle de acesso ao meio baseado no protocolo ALOHA (do inglês, *Additive Links On-Line Hawaii Area*), possibilitando que os dispositivos finais (*end devices*) se comuniquem com um *gateway* através da modulação LoRa (AUGUSTIN *et al.*, 2016).

Uma rede LoRaWAN é composta por dispositivos finais (que podem ser sensores, atuadores etc); pelos *gateways*, que fazem a interface entre os dispositivos finais e a Internet; e pelos servidores, que realizam processamento, tratamento e análise de dados, entre diversas outras ações até a aplicação final. Na Figura 5 é possível observar que a topologia utilizada em uma rede LoRaWAN é estrela e que a comunicação entre os *gateways* e os dispositivos finais é feita com o LoRa. Por outro lado, a comunicação entre os *gateways* e os servidores é estabelecida por meio de uma rede IP (*Internet Protocol*).

Camada Física

Na camada física, a tecnologia LoRa está presente por meio da sua modulação, que foi patenteada pela *Semtech Corporation*. A modulação LoRa é baseada na técnica *Chirp Spread Spectrum* (CSS), que é caracterizada pelo espalhamento espectral do sinal a ser transmitido em determinada faixa de frequências $[f_{low}, f_{high}]$, gerando um sinal denominado

Figura 5: Topologia de uma rede LoRaWAN.



Fonte: Adaptado de Odunlade (2019).

Chirp (*Compressed High Intensity Radar Pulse*) (AUGUSTIN *et al.*, 2016). A técnica de modulação CSS proporciona diversas vantagens ao enlace de comunicação quando comparada a técnicas como modulação em amplitude ou em frequência, sendo essas vantagens dadas por uma maior imunidade a ruídos e interferências (QARAQE *et al.*, 2020), além de maior alcance de transmissão e alta eficiência energética (ELSHABRAWY; ROBERT, 2018).

Um sinal *Chirp* não modulado possui amplitude constante e sua frequência varia em toda a largura de banda ($BW = f_{high} - f_{low}$) por um determinado período de tempo ($T_S =$ Tempo de símbolo). Nesse sinal, a frequência varia linearmente no tempo e tem-se dois tipos de sinais *Chirp*:

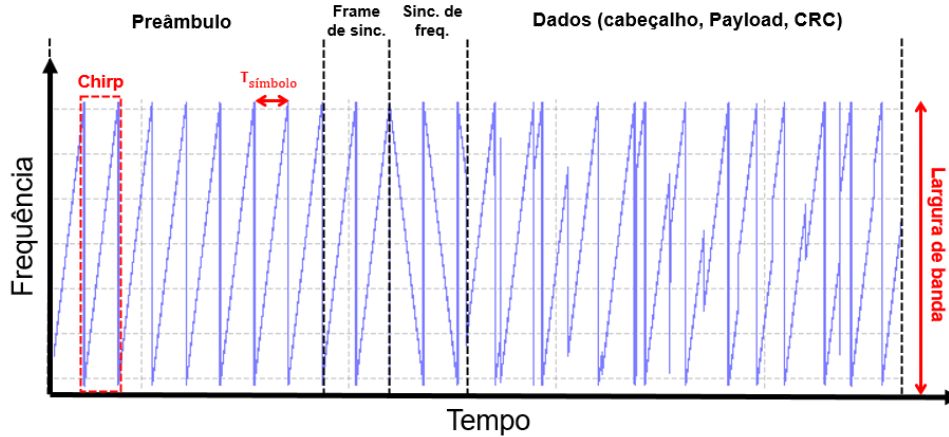
- **Up-Chirp**: Frequência aumenta (f_{low} para f_{high})
- **Down-Chirp**: Frequência diminui (f_{high} para f_{low})

Alguns parâmetros caracterizam a modulação LoRa, a saber: Largura de banda (BW), Fator de espalhamento (SF) e Taxa de código (*Code Rate* - CR).

Um símbolo LoRa, por sua vez, cobre toda a largura de banda (BW) e pode codificar SF bits de dados, em que $SF = 6$ até 12. Um símbolo é composto por um *chirp* crescente que, quando atinge uma frequência dentro da banda determinada pelo dado a ser transmitido, é deslocado para a frequência mínima e mantém o comportamento crescente com mesma

inclinação. Na Figura 6 é ilustrado um exemplo de sinal modulado em LoRa e mostra alguns pontos de descontinuidade, sendo a posição desta descontinuidade na frequência responsável pela codificação do dado transmitido (AUGUSTIN *et al.*, 2016).

Figura 6: Exemplo de pacote de dados modulado em LoRa.



Fonte: Adaptado de Rodrigues *et al.* (2021).

Por meio da Equação 1 é possível obter a relação entre o tempo de símbolo (T_s) e a largura de banda (BW). A partir desta expressão, é possível perceber que, quando o fator de espalhamento (SF) é incrementado, o tempo de símbolo é multiplicado por dois, ou seja, um sinal LoRa com SF = 7 possui o dobro do tempo de símbolo de um sinal LoRa com SF = 6, e assim por diante.

$$T_s = \frac{2^{SF}}{BW}. \quad (1)$$

A Equação 2 expressa a taxa de bits da modulação LoRa em função dos parâmetros de largura de banda, fator de espalhamento e taxa de código:

$$R_b = SF \times \frac{BW}{2^{SF}} \times CR. \quad (2)$$

2.2.2 Sigfox

Uma rede Sigfox, que foi a primeira tecnologia LPWAN proposta para a indústria de IoT e foi desenvolvida pela empresa francesa Sigfox (CENTENARO *et al.*, 2016), apresenta uma camada física que utiliza modulação *Ultra Narrow Band* (UNB). Entretanto,

não há documentação da operação da sua camada de rede, por se tratar de segredo industrial (CENTENARO *et al.*, 2016), o que se torna uma questão relevante em se tratando de estudos acadêmicos sobre a rede e reprodução/simulação de resultados. Uma rede Sigfox opera de maneira similar a uma operadora de telefonia celular para a indústria de IoT (RAZA; KULKARNI; SOORIYABANDARA, 2017), ou seja, existem custos de serviços para que seus usuários usem a rede. A cobertura ou alcance de redes Sigfox, em zonas urbanas, é entre 3 km e 10 km e em zonas rurais entre 30 km a 50 km (CENTENARO *et al.*, 2016)(RAZA; KULKARNI; SOORIYABANDARA, 2017), atuam em banda ISM (868 ou 915 MHz), sua taxa de comunicação é em torno de 100 bps e pode ter até 1.000.000 de nós por *gateway*.

2.2.3 NB-IoT

NB-IoT (*Narrow Band IoT*) é um padrão desenvolvido pela 3GPP (*The 3rd Generation Partnership Project*), que é um organismo internacional de padronização das telecomunicações. A operação do NB-IoT é realizada por operadores de telecomunicações e é uma extensão da infraestrutura de redes celulares 4G (TADAYONI; HENTEN; FALCH, 2017) ou prestadores de serviços 4G LTE, como, por exemplo, Verizon e AT&T nos Estados Unidos. A taxa de transferência de dados pode chegar a 234,7 kbps (TADAYONI; HENTEN; FALCH, 2017) e pode ter 50.000 dispositivos por célula (RAZA; KULKARNI; SOORIYABANDARA, 2017). Uma característica interessante é que o tempo de vida da bateria que alimenta um rádio NB-IoT pode chegar a 10 anos (RAZA; KULKARNI; SOORIYABANDARA, 2017).

2.2.4 Wi-SUN

A tecnologia Wi-SUN (do inglês, *Wireless Smart Ubiquitous Networks*) é mantida pela Wi-SUN Alliance e consiste em redes de comunicação sem fio que são baseadas no padrão IEEE 802.15.4g e que foram projetadas para serem confiáveis e apresentar baixo consumo de energia (SARAIVA; BISPO; GOMES, 2020). Um ponto relevante desta tecnologia é o fato do padrão de comunicação possibilitar o estabelecimento da rede com integração entre dispositivos inteligentes de fabricantes distintos; dessa maneira, podem ser implementadas redes com as topologias estrela, malha ou híbrida (REBELATO, 2018), tornando a área

de cobertura mais ampla (MOCHIZUKI *et al.*, 2017). O Wi-SUN adota um esquema de modulação Gaussiana FSK (GFSK), opera na faixa de frequências não licenciadas, apresenta baixa latência quando comparada a outras tecnologias LPWANs e possui uma taxa de transmissão em torno de 300 kbps (MOCHIZUKI *et al.*, 2017; Wi-SUN Alliance, 2021).

3 Simuladores de Rede

As RSSFs possibilitam implementar diversas aplicações no contexto atual de conectividade; entretanto, mesmo diante dos avanços tecnológicos que favoreceram seu desenvolvimento, essas redes ainda possuem restrições em relação à capacidade de processamento, memória e principalmente em relação ao consumo de energia (MARTINS *et al.*, 2017). Além do mais, as RSSFs geralmente são compostas por um número considerável de dispositivos, sendo sua implementação um processo que pode demandar alto custo e tempo de implantação, dependendo do tamanho da rede (AGUIAR, 2018).

Nesse contexto, as ferramentas de simulação computacional são de grande importância no âmbito de projeto e análise de RSSF, permitindo otimizar aspectos tais como roteamento, segurança e consumo de energia, dentre outros (MARTINS *et al.*, 2017). Assim, os simuladores de rede possibilitam não só obter condições otimizadas de funcionamento antes da implementação física da rede, como também realizar a validação de novas propostas de protocolos por meio de experimentos computacionais (AGUIAR, 2018; HEIDEMANN; MILLS; KUMAR, 2001).

Para que o funcionamento de uma RSSF seja simulado por meio de um *software*, são necessários modelos computacionais capazes de representar o comportamento dos elementos que compõem a rede e também a maneira como estes dispositivos se comunicam. Adicionalmente, as ferramentas também devem ser capazes de modelar condições ambientais que podem impactar na transmissão de dados (MARTINS *et al.*, 2017).

Diante do exposto, neste capítulo são descritos alguns dos principais *softwares* de simulação voltados para redes, suas principais funcionalidades e características.

3.1 Network Simulator (NS-2)

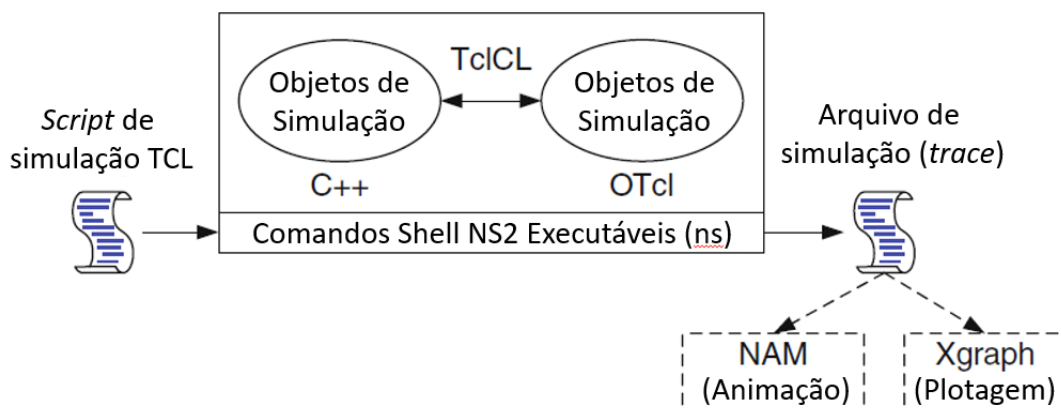
O *software Network Simulator* versão 2 (NS-2) é uma ferramenta de simulação baseada em eventos discretos voltada para pesquisas de redes e que possui código aberto. As simulações do NS-2 são baseadas em modelos computacionais desenvolvidos na linguagem de programação C++ (NSNAM, 2020).

Este simulador foi originalmente desenvolvido para sistemas Unix e proporciona um ambiente de simulação que suporta vários tipos de protocolos, tais como TCP (*Transmission*

Control Protocol), IP, UDP (*User Datagram Protocol*), FTP (*File Transfer Protocol*), entre outros. Podem ser realizadas simulações utilizando taxas de tráfego constantes (*Constant Bit Rate* - CBR) ou variáveis (*Variable Bit Rate* - VBR) (ROCHOL *et al.*, 2003). Além disso, o NS-2 também utiliza a linguagem de comandos de ferramenta orientada a objetos (OTCL), extensão da linguagem TCL (*Tool Command Language*), como uma interface de comando e configuração, e fornece dados sobre o consumo de energia dos nós sensores (AGUIAR, 2018).

As simulações do NS-2 geram um arquivo de saída com os dados obtidos, chamado de *trace*, sendo possível carregar este arquivo em um *software* externo, tal qual o *Xgraph*, para realização de cálculos e obtenção de gráficos. Os cenários de simulação do simulador NS-2 podem ser visualizados por meio do módulo de interface gráfica *Network Animator* (NAM) (ROCHOL *et al.*, 2003). A partir deste módulo é possível visualizar a topologia de rede graficamente e o número de pacotes por *link* (BUCHHEIM, 2002). Na Figura 7 é ilustrada a arquitetura básica do NS-2.

Figura 7: Arquitetura do *software* de simulação de redes NS-2.



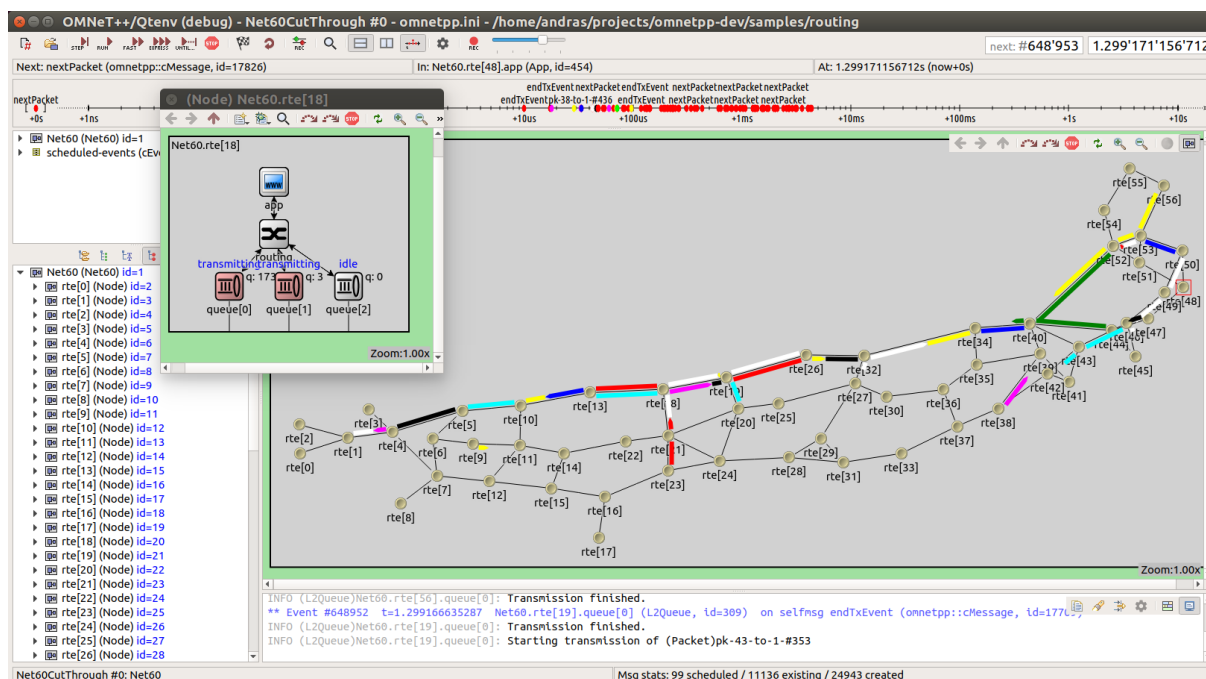
Fonte: (PADIYA, 2013).

3.2 Objective Modular Network Testbed (OMNeT++)

Seguindo a mesma linha do *Network Simulator 2*, o *Objective Modular Network Testbed* (OMNeT++) é uma ferramenta de simulação baseada em eventos discretos e voltada para a modelagem de redes de comunicação, multiprocessadores e sistemas distribuídos (VARGA; HORNIG, 2008).

O OMNeT++ possui como característica uma estrutura modular e extensível, proporcionando ao usuário uma interface gráfica de fácil utilização (OPENSIM, 2019), como exposta na Figura 8.

Figura 8: Interface gráfica do simulador OMNeT++.



Fonte: (OPENSIM, 2019).

Os componentes do OMNeT++ são implementados na linguagem de programação C++ e a interligação entre os módulos é descrita utilizando uma linguagem de alto nível denominada de NED (*Network Definition*). A linguagem NED é própria do OMNeT++ e geralmente é utilizada para descrição das topologias de simulação e hierarquia dos módulos (AGUIAR, 2018).

Existem diversos trabalhos desenvolvidos na literatura utilizando simulações no OMNeT++. Um exemplo é o trabalho proposto por Zangeneh e Ghazvini (2017), no qual é proposto um algoritmo de agrupamento (*clustering*) dinâmico baseado em colônias artificiais. Neste trabalho, os autores mesclam algoritmos genéticos com um algoritmo de colônia de abelhas artificiais visando a otimização de aspectos relacionados ao consumo de energia de uma RSSF.

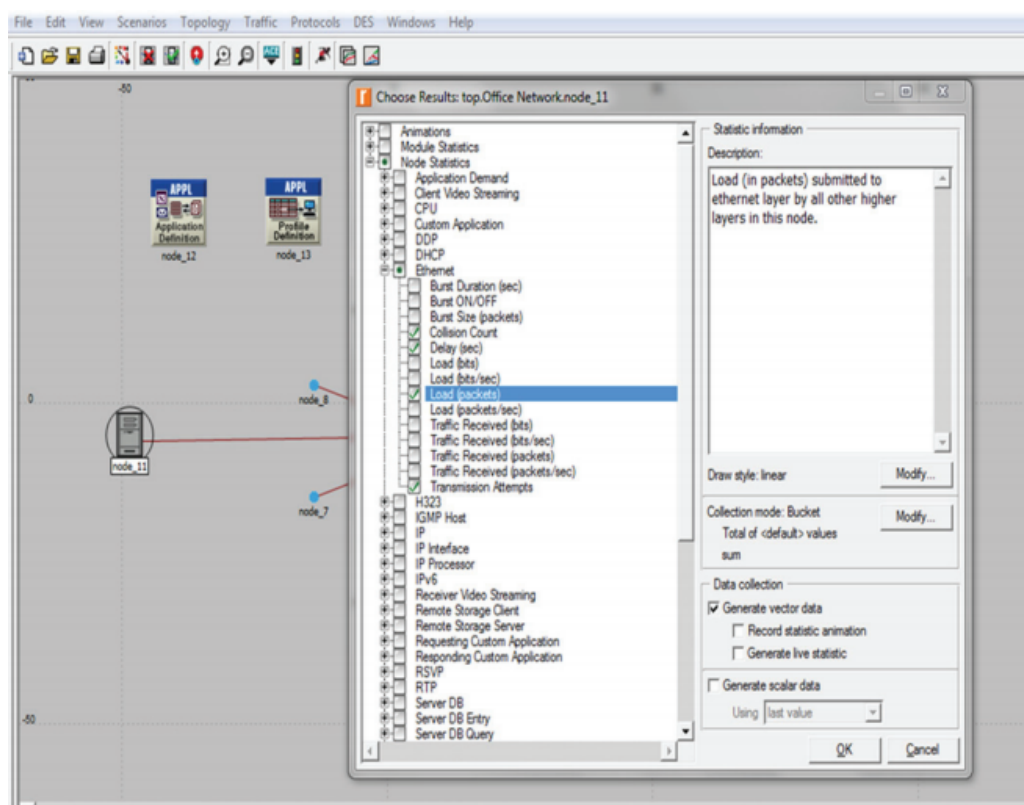
Em Aguiar (2018) é apresentada uma análise de confiabilidade de uma rede de sensores sem fio linear (RSSFL) baseada na comparação entre resultados experimentais e

de simulação. Neste trabalho foi utilizado o Castalia, que é um *framework* baseado no OMNeT++. Nas simulações foram obtidas métricas tais como Taxa de Erro de Pacote (TEP) e Indicador de Intensidade do Sinal Recebido (RSSI).

3.3 Riverbed Modeler

O *Riverbed Modeler Suite* consiste em um conjunto de ferramentas comerciais baseadas em eventos discretos que são direcionadas para projeto e modelagem de redes de telecomunicações. Este *software*, anteriormente conhecido como *OPNET Modeler*, também está disponível em uma versão acadêmica com algumas limitações de uso. O *Riverbed* proporciona um ambiente gráfico, ilustrado na Figura 9, para criação e análise das simulações (ROZORINOV *et al.*, 2017).

Figura 9: Interface gráfica do simulador *Riverbed Modeler*.

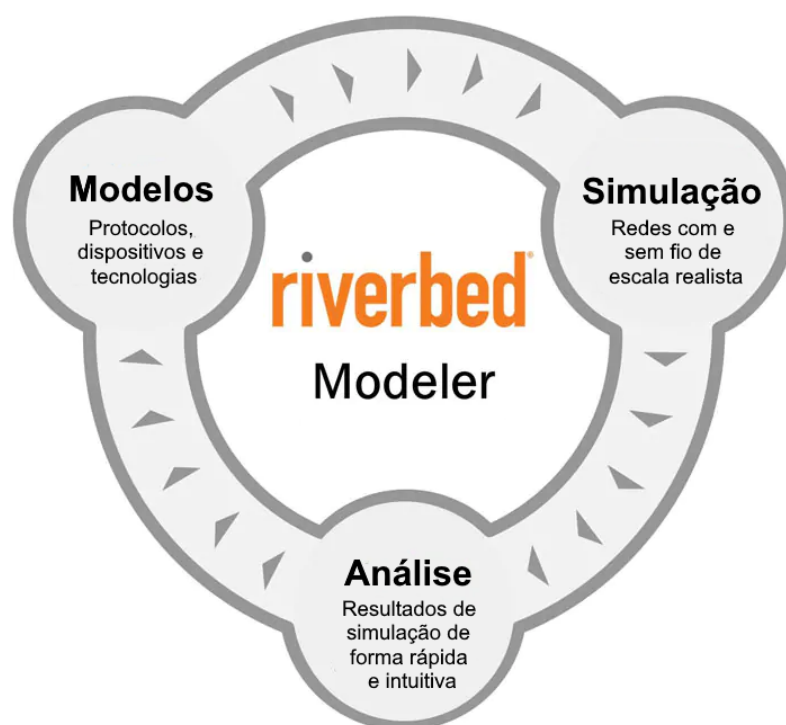


Fonte: (ROZORINOV *et al.*, 2017).

Muitos protocolos e algoritmos de redes estão disponíveis na biblioteca de modelos do *Riverbed* como, por exemplo, alguns protocolos de roteamento dinâmico: RIP (*Routing Information Protocol*), TORA (*Temporally Ordered Routing Algorithm*) etc. Além disso,

também é possível realizar simulações com protocolos da camada de transporte, tais como TCP e UDP (ROZORINOV *et al.*, 2017). O fluxo de funcionamento do *software* está ilustrado na Figura 10 e é dividido em 3 etapas. A primeira etapa refere-se à modelagem dos dispositivos e protocolos, seguida da etapa de simulação (redes com e sem fio de escala realista). Por fim, na terceira etapa, os resultados de simulação podem ser analisados por meio de gráficos (RIVERBED, 2020).

Figura 10: Fluxo de funcionamento do simulador *Riverbed Modeler*.



Fonte: Adaptado de RIVERBED (2020).

Também é possível, por meio dessa ferramenta, implementar e testar novos protocolos e abordagens relacionadas ao desenvolvimento de redes, sendo sua programação realizada na linguagem conhecida como proto-C (BAYRAKDAR, 2020).

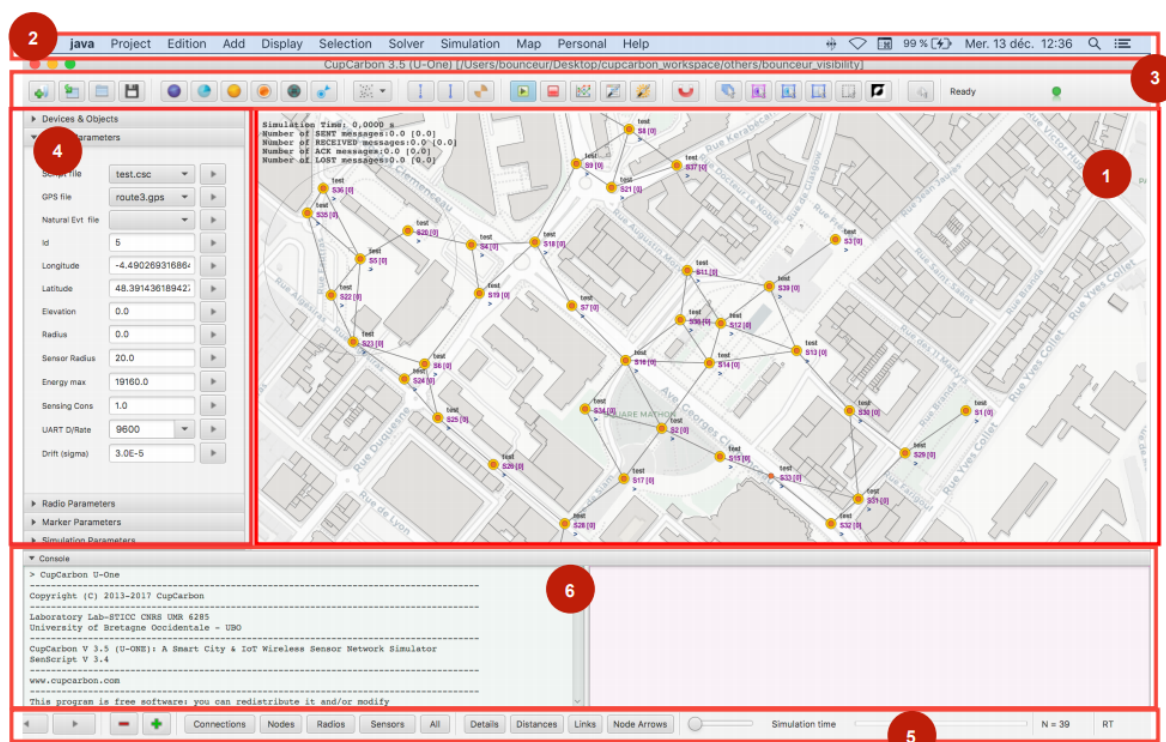
3.4 Cupcarbon

O *Cupcarbon* é um simulador de redes voltado para RSSF que foi desenvolvido em 2014 (BOUNCEUR; MEHDI; LOUNIS, 2014; MEHDI *et al.*, 2014). Trata-se de um *software* de código aberto implementado em Java e orientado para simulações no contexto de cidades inteligentes, RSSF e IoT (BOUNCEUR *et al.*, 2018). Essa ferramenta tem como objetivo

permitir aos projetistas de redes a depuração e validação de algoritmos distribuídos em cenários que podem incluir eventos tais como incêndios, monitoramento de gases, etc. (VIRTUALYS *et al.*, 2019).

A interface gráfica do *Cupcarbon* está ilustrada na Figura 11, com destaque para as suas principais funcionalidades (VIRTUALYS *et al.*, 2019): (1) Mapa, (2) Barra de menu, (3) Barra de ferramentas, (4) Menu de parâmetros, (5) Barra de estado e (6) Console.

Figura 11: Interface gráfica do simulador *Cupcarbon*.

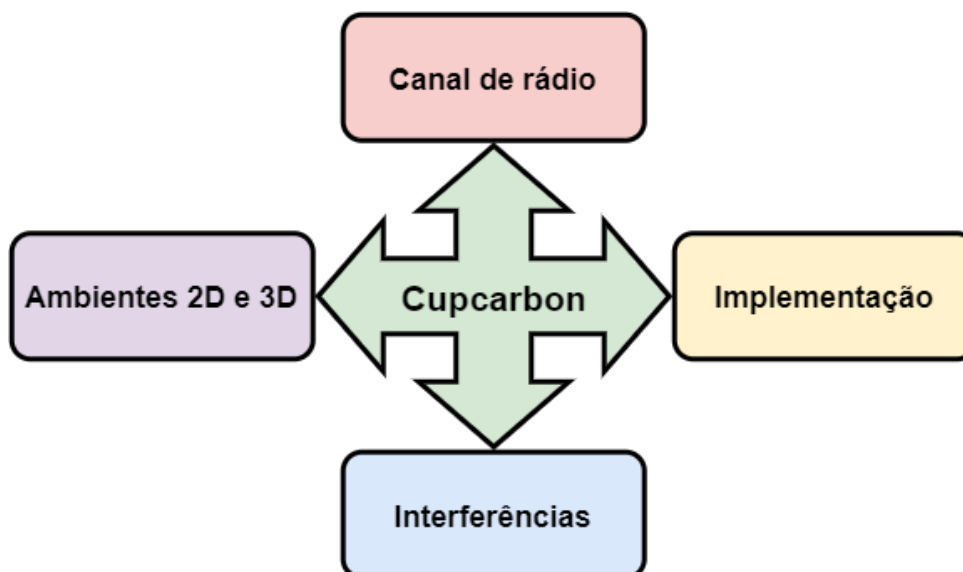


Fonte: (VIRTUALYS *et al.*, 2019).

Na Fig. 12 são ilustrados os quatro principais blocos que compõem a arquitetura do simulador *Cupcarbon*. O primeiro bloco é denominado canal de rádio e abrange dois tipos de modelos de propagação de rádio frequência (RF), sendo o primeiro modelo chamado de árvore de visibilidade (do inglês, *visibility tree*), o qual possibilita estimar a atenuação do canal sem fio e a resposta ao impulso do canal diante de um grande número de receptores. O segundo modelo é um traçado de raios 3D executado em conjunto com o algoritmo de Monte-Carlo e é capaz de explorar o comportamento difuso das superfícies refletoras (BOUNCEUR *et al.*, 2018).

Um ponto de destaque para o *Cupcarbon* é a facilidade de posicionar e visualizar a

Figura 12: Arquitetura do simulador *Cupcarbon*.



Fonte: Adaptado de Bounceur *et al.* (2018).

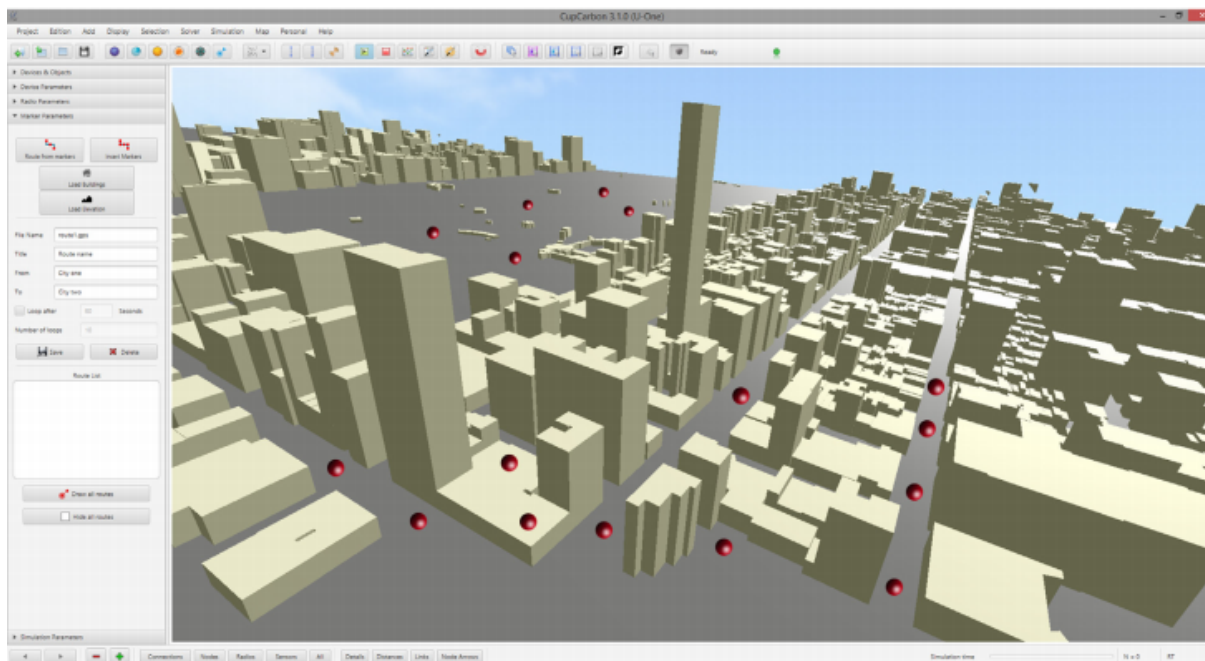
distribuição dos nós de uma RSSF diretamente em localizações específicas de um mapa real. Assim, a topologia de rede pode ser definida na interface gráfica do simulador e pode ser composta por nós sensores, dispositivos móveis, *gateways* e nós IoT (XLIM *et al.*, 2020).

Nesse contexto, o segundo bloco da arquitetura do *Cupcarbon* consiste em um ambiente gráfico em forma de mapas de duas ou três dimensões que podem ser exibidos no *Open Street Maps* (OSM) ou no *Google Maps* (XLIM *et al.*, 2020), como exemplificado na Fig. 13. O ambiente de duas dimensões, ou ambiente 2D, auxilia em atividades de depuração e validação das simulações (BOUNCEUR *et al.*, 2018), enquanto que o ambiente 3D fornece uma maneira mais realista de realizar as simulações, levando em consideração fatores tais como a elevação do solo, edifícios e outros objetos (PHAM *et al.*, 2020).

O bloco de interferências, por sua vez, pode ser dividido em duas partes: camada física e modelos de interferência. Estão integrados na categoria de camada física do *Cupcarbon* modelos realistas de banda base para as seguintes tecnologias de comunicação sem fio (BOUNCEUR *et al.*, 2018):

- Wi-Fi;
- Zigbee;

Figura 13: Exemplo de exibição de mapa em três dimensões no *Cupcarbon*.



Fonte: (PHAM *et al.*, 2020).

- LoRa.

Deste modo, os modelos da camada física possibilitam avaliar nas simulações a qualidade de um enlace de comunicação baseado em condições de transmissão que consideram o canal de rádio e a codificação dos dados com alta precisão (PHAM *et al.*, 2020).

Em redes com alta densidade de dispositivos, a interferência pode ter um impacto significativo na qualidade das transmissões e o bloco de interferências do *Cupcarbon* suporta modelos gaussianos e α -estáveis (PHAM *et al.*, 2020). Bounceur *et al.* (2018) ilustram um exemplo de simulação utilizando um dos modelos de interferência do *Cupcarbon*. Neste exemplo, a transmissão de um pacote Zigbee é corrompida por fontes de interferências Bluetooth e Wifi.

Por fim, a arquitetura do *Cupcarbon* possui uma estrutura de implementação desenvolvida de maneira modular, visando simplificar a substituição e customização de alguma parte específica do simulador. Dessa maneira, torna-se viável simular uma RSSF de acordo com características muito específicas, implementar um mesmo módulo com diferentes propriedades e alterar partes da arquitetura mantendo as demais inalteradas (BOUNCEUR *et al.*, 2018).

A programação dos dispositivos pode ser realizada na linguagem de *script* denominada *Senscript*, proposta por Virtualys *et al.* (2019), também sendo possível gerar códigos para a plataforma de *hardware* Arduino a partir dos algoritmos desenvolvidos. Os nós do tipo IoT podem ser programados em Python a partir da versão 5.0 do *Cupcarbon*, propiciando criar simulações com integrações entre dispositivos reais (ESP32) e virtuais, sendo a comunicação estabelecida segundo o protocolo MQTT (do inglês, *Message Queuing Telemetry Transport*) (XLIM *et al.*, 2020). Além do mais, uma importante característica do *Cupcarbon* é a possibilidade de obter o consumo de energia para cada nó sensor de acordo com dois modelos de consumo: o clássico e o modelo de Heinzelman.

A partir do panorama apresentado para os simuladores de rede neste capítulo, foi escolhido o *software Cupcarbon* para realizar as simulações deste trabalho de pesquisa. Esta escolha baseou-se no fato do *Cupcarbon* ser um *software* multiplataforma de código aberto, com interface gráfica de fácil utilização, aplicável no contexto de cidades inteligentes e que proporciona uma modelagem para o consumo de energia dos dispositivos da rede embasada em publicações científicas relevantes (HEINZELMAN; CHANDRAKASAN; BALAKRISHNAN, 2000; HEINZELMAN; CHANDRAKASAN; BALAKRISHNAN, 2002). Além do mais, por meio do *Cupcarbon* é possível realizar simulações utilizando a tecnologia de comunicação sem fio LoRa, a qual está relacionada ao contexto de pesquisa deste trabalho.

4 Modelos de Perda de Propagação

As ferramentas de simulação de rede dispõem de diversos modelos computacionais, seja para os componentes e dispositivos, para as condições do ambiente, mobilidade ou até mesmo para a propagação dos sinais transmitidos. Em uma comunicação sem fio, o meio de transmissão consiste no canal de rádio entre o transmissor e o receptor. Dessa forma, para simular uma RSSF de maneira adequada é necessário que os simuladores de rede possuam modelos para os canais de rádio que levem em consideração efeitos que interfiram na comunicação entre os dispositivos, como, por exemplo, existência de obstáculos, vegetação, interferências etc. (AGUIAR, 2018).

A potência do sinal recebido varia em função da distância entre o transmissor e o receptor, levando em conta efeitos como a perda de propagação e o sombreamento (AGUIAR, 2018). Os modelos de perda de propagação são equações matemáticas utilizadas para estimar a atenuação entre transmissor e receptor de modo a obter o valor da potência dos sinais recebidos de acordo com condições específicas (frequência, ganho da antena, distância, etc.) (ONIFE *et al.*, 2020), enquanto que os sombreamentos podem ter seu comportamento modelado por uma variável aleatória com distribuição lognormal em função de obstáculos existentes entre transmissor e receptor (AGUIAR, 2018).

Existem na literatura basicamente três tipos de modelos de perda de propagação: os empíricos, semideterminísticos e determinísticos (SHARMA; SHARMA; SAI, 2018). Os modelos empíricos são obtidos a partir de resultados experimentais e observações obtidos na área de interesse e geralmente costumam ser mais precisos, porém limitados à situação para a qual foram desenvolvidos (SILVA, 2014). Por outro lado, os modelos determinísticos (ou teóricos) podem ser obtidos a partir das equações da teoria eletromagnética, em que é possível calcular a potência do sinal recebido para diferentes tipos de ambientes, fornecendo uma boa aproximação das perdas de propagação e das características do canal (SHARMA; SHARMA; SAI, 2018; BARIZON, 2004). Os modelos semideterminísticos (ou semiempíricos), por sua vez, possuem como característica a combinação da teoria com os resultados experimentais obtidos nas medições (BARIZON, 2004).

Tendo em vista que cada modelo possui suas especificidades e que nenhum deles é adequado de maneira universal para todas as situações e ambientes, pode-se concluir que a sua aplicabilidade também é limitada a determinadas situações (SILVA, 2014). Sendo

assim, os simuladores de redes podem disponibilizar diferentes modelos de propagação.

No *software Network Simulator 2* (NS-2), por exemplo, estão disponíveis três modelos de propagação para simulação do canal sem fio: modelo de espaço livre, modelo de dois raios e modelo de sombreamento (PENG, 2015). No *Cupcarbon*, por sua vez, estão integrados dois modelos de propagação, sendo o primeiro modelo chamado de árvore de visibilidade e o segundo é baseado em um traçado de raios 3D executado em conjunto com o algoritmo de Monte-Carlo (BOUNCEUR *et al.*, 2018).

Neste capítulo serão expostos alguns dos principais modelos de propagação disponíveis na literatura e suas principais características.

4.1 Modelo de Espaço Livre

O conceito de espaço livre considera um canal de comunicação sem a presença de obstáculos, de maneira que a propagação eletromagnética não sofra influência de efeitos tais como absorção, reflexão, refração ou difração. Nesta situação, considera-se que a superfície do solo está muito longe ou que o seu coeficiente de reflexão é desprezível, caracterizando uma situação ideal de comunicação (SKLAR, 2009); no entanto, isto representa uma boa aproximação para linha de visada direta operando em altas frequências e sem efeito da reflexão do solo (CASTELLANOS, 2003).

Dessa forma, a potência do sinal recebido depende apenas da distância entre os dispositivos e da frequência de transmissão (SKLAR, 2009). Matematicamente, o modelo de perda de propagação de espaço livre para uma linha de visada direta pode ser expresso como (ONIPE *et al.*, 2020):

$$P_L(dB) = 32,44 + 20\log_{10}(f) + 20\log_{10}(d), \quad (3)$$

em que f representa a frequência de transmissão em MHz e d a distância entre o transmissor e o receptor em quilômetros.

4.2 Modelo de Okumura

Um dos modelos mais utilizados para predição de cobertura em áreas urbanas é o modelo de Okumura, que consiste em um modelo empírico que foi proposto a partir

de resultados experimentais obtidos no Japão (OKUMURA *et al.*, 1968). Este modelo contempla diversos tipos de ambiente (urbano, suburbano e rural) e leva em consideração parâmetros como o tipo de ambiente e a características do terreno (MARDENI; PEY, 2010). Os experimentos foram realizados na faixa de frequências compreendida entre 150 e 1920 MHz e também foram levadas em consideração distâncias entre 1 e 100 km, assim como uma variação na altura da antena de transmissão de 30 a 1000 m (ONIFE *et al.*, 2020).

Diante desse contexto, Okumura elaborou um conjunto de curvas que relacionam a atenuação média em relação ao espaço livre para um terreno quase plano, sendo a altura da antena do transmissor igual a 200 m e do receptor 3 m (MARDENI; PEY, 2010). A partir destas curvas, também é possível realizar ajustes no modelo de acordo com o tipo de ambiente em que estão sendo realizadas as transmissões. O modelo de Okumura é dado por (ONIFE *et al.*, 2020):

$$P_L = P_{LF} + A_{mu}(f, d) - G(h_t) - G(h_r) - G_{AREA}, \quad (4)$$

em que P_{LF} corresponde à perda de propagação em espaço livre, A_{mu} é a atenuação média relativa, $G(h_t)$ é o fator de ganho da altura da antena transmissora, $G(h_r)$ representa o fator de ganho da altura da antena receptora e G_{AREA} é o ganho devido ao tipo do ambiente.

Os fatores de ganho das antenas do receptor e transmissor podem ser obtidos a partir das equações a seguir (ONIFE *et al.*, 2020):

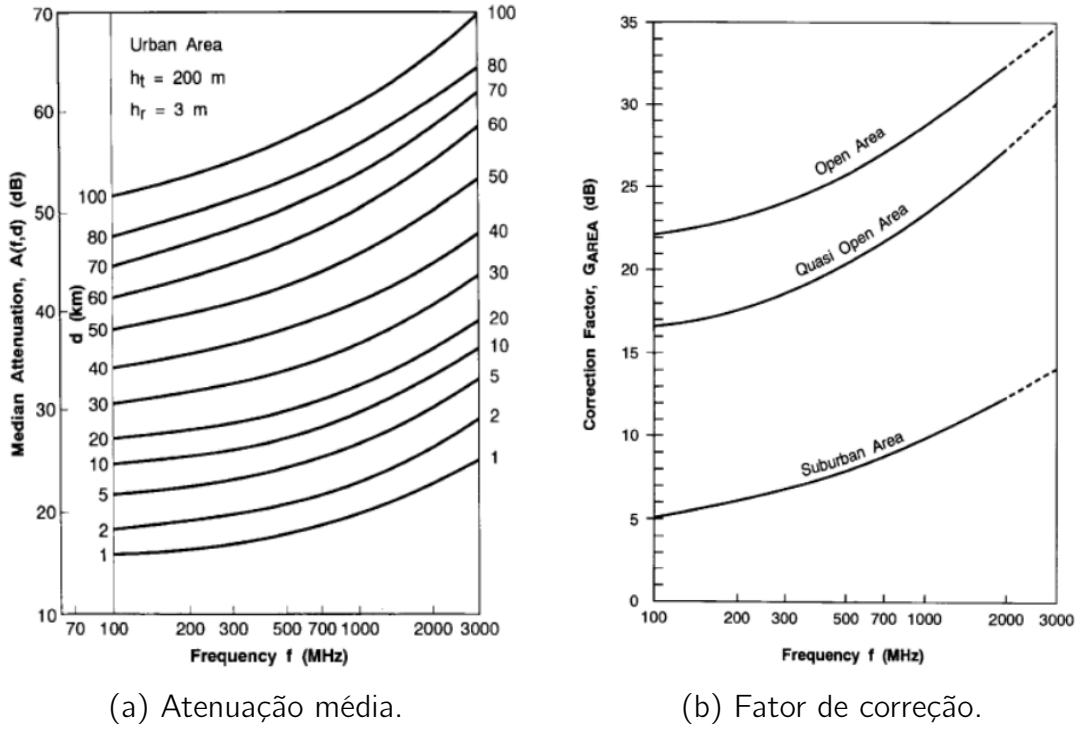
$$G(h_t) = 20 \log \left(\frac{h_t}{200} \right), \quad 1000 \text{ m} > h_t > 30 \text{ m} \quad (5)$$

$$G(h_r) = 10 \log \left(\frac{h_r}{3} \right), \quad h_r \leq 3 \text{ m} \quad (6)$$

$$G(h_r) = 20 \log \left(\frac{h_r}{3} \right), \quad 3 \text{ m} < h_r \leq 10 \text{ m}. \quad (7)$$

Estão ilustradas nas Figs. 14.a e 14.b as curvas do modelo de Okumura, por meio das quais é possível obter os parâmetros A_{mu} e G_{AREA} da equação.

Figura 14: Curvas do modelo de Okumura.



(a) Atenuação média.

(b) Fator de correção.

Fonte: (OKUMURA *et al.*, 1968).

4.3 Modelo de Okumura-Hata

Este modelo compreende uma extensão do que foi desenvolvido por Okumura. Enquanto que no modelo clássico de Okumura são fornecidas curvas a partir dos resultados experimentais para obtenção da atenuação média e dos fatores de correção, o modelo de Okumura-Hata, também conhecido como modelo de Hata, fornece equações matemáticas empíricas que possibilitam calcular os resultados dos gráficos do modelo clássico para diversos parâmetros. Dessa forma, torna-se mais simples e viável a implementação deste modelo nos *softwares* de simulação (HARINDA *et al.*, 2019). O modelo de Hata inclui os efeitos de difração, reflexão e espalhamento causados pelas estruturas da cidade (MARDENI; PEY, 2010) e compreende cenários do tipo urbano, suburbano e rural. Para áreas urbanas, o modelo de Okumura-Hata pode ser expresso por (SANTOS *et al.*, 2020):

$$P_L = 69,55 + 26,16 \log(f_c) - 13,82 \log(h_t) - a(h_r) + [44,9 - 6,55 \log(h_t)] \log(d), \quad (8)$$

em que f_c corresponde à frequência da portadora em MHz, h_t é a altura da antena da

estação base (transmissão) em metros, h_r é a altura da antena da estação móvel (receptora) em metros, d é a distância entre o transmissor e o receptor em quilômetros e $a(h_r)$ é um fator de correção para altura efetiva da antena da estação móvel e depende do tipo de ambiente.

O fator de correção pode ser obtido para pequenas e médias cidades por meio da expressão (ONIFE *et al.*, 2020):

$$a(h_r) = (1,1 \log(f_c) - 0,7)h_r - (1,56 \log(f_c) - 0,8)dB. \quad (9)$$

Para grandes cidades, o fator de correção é dado por:

$$a(h_r) = 8,29(\log(1,54h_r))^2 - 1,1dB, \quad f_c \leq 300 \text{ MHz} \quad (10)$$

$$a(h_r) = 3,2(\log(1,75h_r))^2 - 4,97dB, \quad f_c \geq 300 \text{ MHz} \quad (11)$$

4.4 Modelo COST 231

Da mesma maneira que o modelo de Hata corresponde a um complemento do modelo de Okumura, o modelo COST 231 consiste em uma extensão do modelo de Hata e foi estabelecido pelo programa COST (em inglês, *European Cooperation in Science and Technology*) para ser utilizado na faixa de frequências compreendida entre 500 e 2000 MHz (MALOKU *et al.*, 2019). O modelo COST 231 contém correções para ambientes rurais, suburbanos e urbanos (SHARMA; SHARMA; SAI, 2018) e pode ser calculado por meio das expressões apresentadas a seguir (HARINDA *et al.*, 2019):

$$P_L = 46,3 + 33,9 \log(f_c) - 13,82 \log(h_t) - a(h_r) + (44,9 - 6,55 \log(h_t) \log(d)) + c_m, \quad (12)$$

em que f_c é a frequência da portadora em MHz, h_t e h_r correspondem às alturas das antenas da estação base e da estação móvel, respectivamente, e d representa a distância entre o transmissor e o receptor em quilômetros. O parâmetro c_m está associado ao tipo de ambiente e possui o valor de 0 dB para cenários rural e suburbano e 3 dB para área urbana (MALOKU *et al.*, 2019). A variável $a(h_r)$ pode ser obtida para áreas urbanas pela seguinte expressão (HARINDA *et al.*, 2019):

$$a(h_r) = 3,20(\log(11,75h_r))^2 - 4,97, \text{ para } f > 400\text{MHz}. \quad (13)$$

Para áreas suburbanas e rurais, o valor da variável $a(h_r)$ é calculado como exposto a seguir:

$$a(h_r) = (1,1\log(f_c) - 0,7)h_r - (1,56\log(f_c) - 0,8). \quad (14)$$

4.5 Modelo de Egli

O modelo de Egli é um modelo de perda de propagação muito utilizado e que foi desenvolvido a partir de resultados experimentais obtidos em medições de sistemas de transmissão de televisão (SILVA, 2014). Este modelo considera uma linha de visada direta entre os dispositivos e é também adequado para situações de comunicação celular em que um dos dispositivos é fixo e o outro móvel. Além disso, também é aplicável em cenários em que a transmissão ocorre em meio a terrenos irregulares, mas sem a presença de vegetação no *link* de comunicação (MARDENI; PEY, 2010). O modelo de Egli pode ser usado para a faixa de frequência compreendida entre 40 e 1000 MHz e para distâncias de até 60 km (MALOKU *et al.*, 2019). A expressão matemática do modelo de Egli está apresentada a seguir (ONIFE *et al.*, 2020):

$$P_L = G_t G_r \left(\frac{h_t h_r}{d^2} \right)^2 \left(\frac{40}{f_c} \right)^2, \quad (15)$$

em que G_t e G_r correspondem, respectivamente, aos ganhos das antenas do transmissor e do receptor, h_t e h_r são as alturas das antenas do transmissor e receptor, d é a distância entre os dispositivos e f_c é a frequência da portadora em MHz.

Diante do que foi apresentado neste capítulo, foi adotado o modelo de Egli para ser implementado no simulador *Cupcarbon*. A justificativa é que se trata de um modelo com equacionamento simples, que compreende a faixa de frequências dos módulos de comunicação LoRa utilizados neste trabalho e que também é adequado para o raio de distância de cobertura da área na qual serão realizadas as simulações. Além disso, também foi levado em consideração o fato do modelo de Egli ser adequado para situações na qual um dispositivo possui mobilidade e o outro permanece fixo. Assim, este procedimento será apresentado com detalhes no capítulo de metodologia.

5 Protocolos de Roteamento para RSSF

Este capítulo apresenta, de maneira geral, algumas noções sobre o processo de roteamento de dados em uma rede de sensores; quais são os principais parâmetros utilizados para definição das rotas ótimas; e também são classificados os principais protocolos de roteamento para redes sem fio *ad hoc*. Adicionalmente, são apresentados alguns protocolos e algoritmos que foram utilizados na etapa de implementação por meio de simulação computacional.

5.1 Visão Geral sobre Roteamento

Diante de uma realidade de dispositivos conectados em rede e trocando informações entre si, torna-se muito importante uma discussão sobre as possibilidades de compartilhamento de dados, especialmente a maneira como esse procedimento é executado. Assim, para que seja possível enviar um dado de um nó sensor até outro dispositivo e esse dado tenha que percorrer dispositivos intermediários, torna-se necessário definir por qual caminho os dados deverão ser encaminhados, caracterizando assim o processo de roteamento.

Dessa forma, o roteamento consiste no processo de avaliação dos caminhos disponíveis a partir de um nó fonte até um determinado nó destino e da determinação de um caminho para encaminhamento dos pacotes de dados na rede (GAWIY *et al.*, 2014).

Pela ótica da abstração de camadas do modelo OSI (*Open System Interconnection*), referência da ISO (*International Organization for Standardization*), o procedimento de roteamento é executado na camada de rede (FARIAS, 2008), por meio de um algoritmo de roteamento, que é responsável pelo encaminhamento dos pacotes de dados (TANENBAUM, 2011).

Os algoritmos de roteamento devem utilizar critérios para definição das melhores rotas, como, por exemplo (FARIAS, 2008):

- **Número de saltos:** este critério se refere à quantidade de nós que um determinado pacote percorre até chegar ao seu destino (*hop count*), mas também poderá se referir à distância em metros entre os nós de origem e destino;
- **Estado de *link*:** ao levar em consideração o critério de estado de *link*, o algoritmo de roteamento poderá tomar a decisão da melhor rota baseado em informações da

rede tais como velocidade, latência, largura de banda etc;

- **Nível de carga de bateria;**
- **Posição geográfica (coordenadas).**

Os principais protocolos de roteamento para redes *ad hoc* sem fio possuem seu funcionamento embasado em algoritmos que utilizam um ou mais dos critérios apresentados acima, dando origem aos algoritmos na categoria vetor-distância e estado de *link* (BHATT; HEMRAJANI, 2013).

5.1.1 Algoritmos Vetor-Distância

Os algoritmos baseados na abordagem de vetor-distância utilizam como referência o algoritmo distribuído de Bellman-Ford (DBF) e recorrem a parâmetros como distância e direção entre as rotas disponíveis para determinação do caminho ótimo (BHATT; HEMRAJANI, 2013). Além disso, as informações sobre os caminhos possíveis para cada destino são obtidas a partir do compartilhamento de tabelas entre nós adjacentes da rede.

Dessa maneira, cada dispositivo mantém armazenada em sua memória uma tabela de roteamento, na qual estão contidas informações sobre os menores caminhos para cada destino (TANENBAUM, 2011). Periodicamente, essas tabelas são compartilhadas via *broadcast* com todos os nós da rede e comparadas com as tabelas atuais; caso alguma atualização seja detectada, o nó identifica a origem da informação e atualiza a sua tabela de roteamento, compartilhando em seguida a nova tabela com os seus respectivos nós vizinhos (FARIAS, 2008). Este compartilhamento periódico de tabelas ocorre mesmo que não existam atualizações na topologia da rede (BHATT; HEMRAJANI, 2013).

5.1.2 Algoritmos de Estado de *link*

Apesar da ideia geral dos algoritmos que utilizam a abordagem vetor-distância ser bastante simples e útil para encontrar o menor caminho até um destino específico de uma rede, é possível que essa abordagem não seja a mais eficiente por apresentar, por exemplo, uma maior latência (TANENBAUM, 2011; FARIAS, 2008).

Por outro lado, os algoritmos de estado de *link* (ou estado de enlace) surgiram com o intuito de levar em consideração, no processo de decisão de rotas para encaminhamento

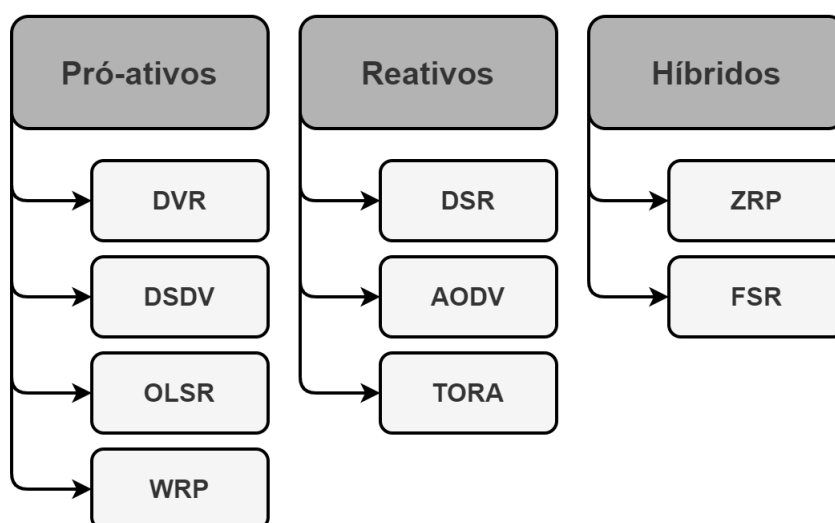
de dados, não apenas a distância entre os nós, mas também características tais como latência média de cada caminho, largura de banda disponível, velocidade etc (BHATT; HEMRAJANI, 2013) - e são implementados embasados no algoritmo de Dijkstra (1959) para cálculo da rota mais curta para cada destino.

5.2 Classificação dos Protocolos para Redes *ad hoc*

Os protocolos de roteamento para redes *ad hoc* sem fio podem ser classificados quanto à sua natureza e propriedades seja com base nas informações compartilhadas, na forma com que são compartilhadas ou mesmo em relação a quando e como as rotas são calculadas (SRIVASTAVA *et al.*, 2014; DEBBARMA; SEN; ROY, 2012). Nesse sentido, alguns exemplos de classificações são: protocolos de estado de *link*, protocolos baseados em vetor-distância, cientes de energia (*energy-aware*), com roteamento de origem, multicaminho, dentre outros (DEBBARMA; SEN; ROY, 2012).

Porém, neste trabalho será dado destaque às categorias de protocolos pró-ativos, reativos e híbridos, que se diferenciam na maneira como operam, de acordo com a estratégia de roteamento (RAHMAN; HASAN; ISLAM, 2012). É possível observar na Figura 15 alguns dos principais protocolos de roteamento de acordo com essas três categorias, sendo alguns deles explicados adiante.

Figura 15: Classificação dos protocolos de roteamento para redes *ad hoc* sem fio.



Fonte: Adaptado de (DEBBARMA; SEN; ROY, 2012).

5.2.1 Pró-ativos

Os protocolos de roteamento pró-ativos, também conhecidos como protocolos *table driven*, compartilham as suas tabelas de roteamento com os nós vizinhos durante a inicialização da rede. Desta forma, todos os dispositivos terão conhecimento acerca dos caminhos para quaisquer nós de destino antes mesmo que essas informações sejam necessárias (PARVATHI, 2012).

Para que seja possível obter com antecedência as informações sobre todos os destinos, e em todas as tabelas de roteamento da rede, é necessário um mecanismo de compartilhamento periódico sobre as rotas. Por isso, os protocolos de roteamento pró-ativos realizam o processo de atualização das tabelas de roteamento não só ao inicializar a rede, mas também sempre que ocorra uma mudança de topologia ou em intervalos fixos. Para tal, é necessário que cada nó sensor da rede se comunique com seus vizinhos e troque informações a respeito dos demais nós.

Segundo Matre e Karandikar (2016), um ponto de destaque nos protocolos pró-ativos é a sua baixa latência em relação aos protocolos das demais categorias, uma vez que a rota do nó de origem até o destino é obtida ou atualizada nas tabelas de roteamento da rede antes que seja necessário enviar o dado.

Alguns dos principais protocolos pró-ativos estão listados a seguir:

- *Distance Vector Routing Protocol (DVR)*;
- *Destination-Sequenced Distance Vector Routing Protocol (DSDV)*;
- *Optimized Link State Routing Protocol (OLSR)*;
- *Fish Eye State Routing Protocol (FSR)*;
- *Cluster Gateway Switch Routing Protocol (CGSR)*;
- *Wireless Routing Protocol (WRP)*.

5.2.1.1 *Distance Vector Routing Protocol (DVR)*

O funcionamento deste protocolo está essencialmente ligado ao algoritmo de roteamento de vetor-distância, também chamado de algoritmo distribuído de Bellman-Ford ou conhecido como *Routing Information Protocol* (RIP) (HEDRICK, 1988), e foi o algoritmo de roteamento original da ARPANET (que antecedeu a Internet). Sendo assim, cada nó sensor de uma rede de comunicação sem fio mantém uma tabela de roteamento com as seguintes informações (MEDHI; RAMASAMY, 2007):

- Endereço de cada destino;
- Métrica ou custo;
- Próximo salto.

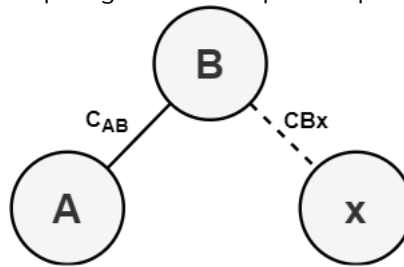
O número de saltos necessários para que um determinado pacote chegue ao seu destino final determina a métrica de custo; ou seja, refere-se ao número de nós que serão percorridos até que a informação seja recebida de fato. Por sua vez, o conhecimento sobre para qual nó encaminhar o pacote de dados primeiro, quando for necessário enviar algo até um determinado destino, é armazenado em uma coluna denominada de próximo salto da tabela de roteamento.

Para uma melhor compreensão, será apresentado um exemplo deste protocolo utilizando a topologia ilustrada na Figura 16. No algoritmo do Quadro 2, está descrito o funcionamento deste exemplo, baseado na descoberta do menor caminho (MEDHI; RAMASAMY, 2007). Supondo que um determinado nó fonte A necessite enviar um pacote de dados para um nó x fora de seu raio de alcance, é possível perceber na Figura 16 que devido à topologia da rede, o dados deverão ser transmitidos primeiro até o nó B para que em seguida sejam encaminhados para o seu destino final, o nó x . A tabela de roteamento inicial do nó A, antes do compartilhamento de dados, está ilustrada no Quadro 1.

Quadro 1: Tabela de roteamento inicial do nó A (Exemplo do protocolo DVR)

Destino	Métrica (custo)	Próximo salto
A	0	A
B	1	B
x	∞	∞

Figura 16: Topologia de exemplo do protocolo DVR.



Fonte: Adaptado de (MEDHI; RAMASAMY, 2007).

Quadro 2: Algoritmo DVR.

Algoritmo 1: Algoritmo do exemplo do protocolo *Distance Vector Routing* (DVR)

Dados: Tabela de roteamento de um nó vizinho B com conexão direta

Resultado: Descoberta dos caminhos mais curtos até cada destino x fora de alcance

```

/* Inicialização das métricas de custo                                     */
1  CAA = 0;
2  CAB = 1;
3  CAx = ∞;
4  for cada nó x fora de alcance do
5      ler o custo do nó x na tabela do nó B ;
6      CAx = min { CAB + CBx }
7  end
  
```

Fonte: Adaptado de (MEDHI; RAMASAMY, 2007).

De acordo com o algoritmo mostrado no Quadro 2, ao inicializar a rede, cada nó sensor deverá preencher a coluna de custos em sua tabela de roteamento levando em consideração alguns critérios. O custo próprio de cada nó (número de saltos até ele mesmo) deverá ser zero, o custo para um nó sensor com conexão direta terá um valor unitário e o custo para os demais nós, fora do raio de alcance, terá um custo inicialmente infinito (∞), indicando que não é possível enviar um pacote de dados diretamente para tais nós.

Em seguida, ao receber durante o intervalo fixo de atualização uma tabela de roteamento de um nó adjacente, será realizada uma comparação direta entre cada linha das tabelas recebida e atual, executando um processo de minimização entre a soma dos custos do caminho até o nó de destino, conforme exemplificado na linha 6 do Algoritmo 1.

Portanto, ao final da execução do algoritmo de roteamento do protocolo DVR, a tabela de roteamento atualizada do nó A deverá ser preenchida como exposto no Quadro 3, sendo possível observar que para enviar um pacote de dados para o nó x , será necessário encaminhá-lo primeiro ao nó B. O Algoritmo 1 também pode ser utilizado para descoberta de caminhos para outros nós que estejam fora do alcance do nó A.

Quadro 3: Tabela de roteamento final do nó A (Exemplo do protocolo DVR)

Destino	Métrica (custo)	Próximo salto
A	0	A
B	1	B
x	2	B

Um ponto a se observar em algoritmos deste tipo é que, como são realizadas constantes trocas de informações sobre as rotas da rede, o gasto energético pode ser considerável, a depender do período de atualização. Outro aspecto impactado devido ao alto volume de dados de controle é a sobrecarga na rede, caso ela seja composta por muitos nós sensores.

Além disso, existem alguns problemas bastante conhecidos para protocolos baseados em vetor-distância, como por exemplo o problema de contagem até o infinito¹ (MANNAN; JAYAVIGNESH, 2017) e a ocorrência de *loops* de roteamento², conforme indicado em (MEDHI; RAMASAMY, 2007). Caso ocorra alguma mudança de topologia na rede, os nós sensores poderão encaminhar atualizações com informações obsoletas e ocasionar encaminhamentos errôneos durante o envio de dados. Existem diversos trabalhos que propõem soluções para contornar estes problemas; um deles será apresentado a seguir.

5.2.1.2 Destination Sequenced Distance Vector (DSDV)

O protocolo *Destination-Sequenced Distance Vector* (DSDV), proposto em 1994 por Charles Perkins (PERKINS; BHAGWAT, 1994), trata-se de um protocolo pró-ativo que é uma extensão do DVR e visa resolver os problemas de contagem ao infinito e *loops* infinitos, conforme mencionado na seção anterior. Segundo Perkins, a principal causa para formação dos *loops* de roteamento é que os nós escolhem seus próximos saltos

¹Problema que ocorre quando há perda de enlace na rede e os roteadores atualizam suas tabelas de roteamento com informações desatualizadas, criando um *loop* infinito com contagem incorreta de saltos.

²Situação que ocorre quando um nó encaminha dados para outro nó, que por sua vez encaminha para o salto seguinte e assim por diante, sendo que ao final, o último nó encaminha os dados de volta para o primeiro nó, criando um tráfego circular.

de maneira distribuída, com base em informações que podem ser obsoletas e incorretas. Assim, o DSDV foi implementado visando dois objetivos principais: manter a simplicidade do algoritmo de Bellman-Ford e resolver o problema dos *loops* de roteamento.

Por se tratar de um protocolo que utiliza como base o mesmo algoritmo do protocolo DVR, o protocolo DSDV mantém em sua tabela de roteamento as mesmas informações, porém adicionando uma nova coluna. Dessa forma, ao utilizar o protocolo DSDV, os nós terão armazenadas em suas tabelas de roteamento as seguintes informações:

- Endereço de cada destino;
- Métrica ou custo;
- Próximo salto;
- Número de sequência.

O número de sequência, proposto no trabalho de Perkins, foi criado justamente para possibilitar uma diferenciação entre as informações obsoletas e as mais atuais. Dessa forma, ao detectar uma perda de conexão, será feito um incremento no número de sequência do nó *offline* e em seguida uma atualização será encaminhada para a rede. As rotinas de atualização do DSDV ocorrem em intervalos fixos, assim como no DVR, compartilhando a tabela de roteamento completa ou por meio de atualizações parciais, as quais contêm apenas as alterações de topologia. Os números de sequência, por sua vez, são geralmente pares, se o *link* estiver presente, e ímpares, se não há conexão (BHATT; HEMRAJANI, 2013). Além do mais, tendo em vista que o algoritmo para determinar o menor caminho utilizado pelo DSDV é o mesmo do protocolo DVR, a métrica de custo é consonante com a já apresentada no Algoritmo 1:

$$C_{ix} = \min\{C_{ij} + C_{jx}\}, \quad (16)$$

em que i representa o nó fonte, j um nó com conexão direta para o nó i e x o nó de destino.

5.2.2 Reativos

Ao contrário dos protocolos da categoria pró-ativo, os protocolos reativos estabelecem rotas para um nó de destino apenas quando necessário, ou seja, quando há pacotes a serem enviados e, portanto, mantêm armazenadas apenas as rotas ativas para os nós de destino em uso. Sendo assim, estes protocolos não compartilham informações ao inicializar a rede, nem possuem mecanismos periódicos de compartilhamento de tabelas de roteamento (MATRE; KARANDIKAR, 2016).

Devido ao fato das informações de roteamento não serem mantidas caso não exista necessidade prévia de comunicação, esta categoria de protocolos também é conhecida como sob demanda e o processo de descoberta de rotas geralmente ocorre por meio de mensagens de inundação (ou *flooding*) (PARVATHI, 2012). Por esta razão, os protocolos reativos apresentam uma maior latência em relação aos pró-ativos, pois necessitam encontrar uma rota no caso de uma demanda de transmissão. Porém, em situações de maiores mudanças na topologia de rede, normalmente estes protocolos possuem um melhor desempenho (MATRE; KARANDIKAR, 2016).

5.2.2.1 *Ad Hoc On-Demand Distance Vector (AODV)*

A abordagem dos algoritmos vetor-distância é utilizada por diversos protocolos de roteamento e novas abordagens continuam sendo propostas, que são baseados em outros já existentes e visam melhorar aspectos frágeis ou otimização de desempenho, por exemplo. Desta forma, o *Ad Hoc On-Demand Distance Vector (AODV)* refere-se a um protocolo de roteamento da categoria dos reativos que possui seu funcionamento baseado no DSDV, tendo sido proposto em 1999 por Charles Perkins, o mesmo autor do DSDV (PERKINS; ROYER, 1999).

Este protocolo realiza a descoberta de rotas apenas quando existe uma demanda de transmissão e o caminho até o destino não é conhecido ou está indisponível, motivo pelo qual apresenta um maior desempenho quando submetido a mudanças de topologia mais constantes, favorecendo um roteamento dinâmico. Além disso, utiliza uma abordagem de comunicação bidirecional, possibilitando desenvolver o mecanismo de descoberta e manutenção de rotas baseado em um modelo de requisições e respostas. Diante disso, os principais tipos de mensagens utilizados pelo AODV estão detalhados abaixo (PERKINS;

ROYER, 1999; FARIAS, 2008):

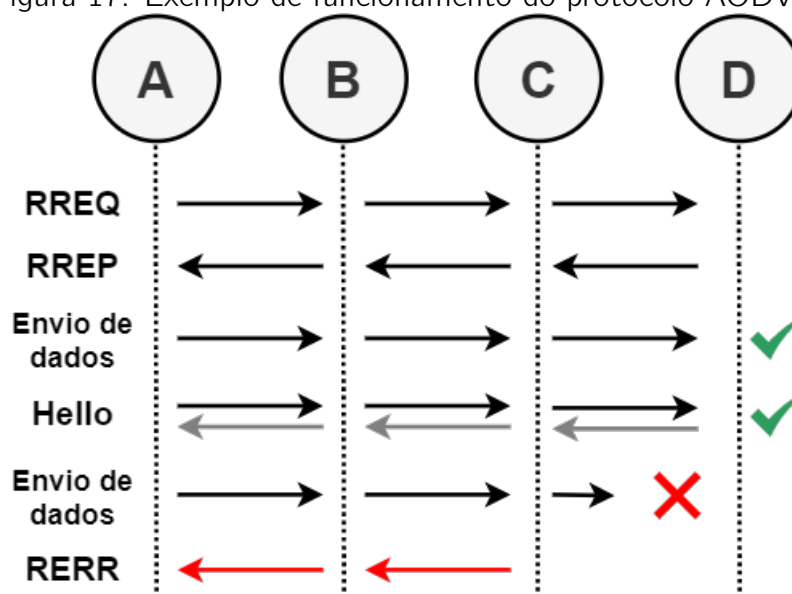
- **RREQ:** a *Route Request* é uma mensagem utilizada pelo AODV para requisitar uma rota até determinado destino. Esta mensagem é enviada via *broadcast* para todos os nós adjacentes;
- **RREP:** sempre que um nó da rede receber uma mensagem RREQ e for o nó de destino, deverá enviar para o nó fonte uma mensagem de resposta (*Route Reply* - RREP) informando o seu número de sequência atual;
- **RRER:** *Route Error* é uma mensagem de erro enviada ao nó fonte sempre que uma determinada rota ficar indisponível. Pode ser enviada durante uma mensagem de manutenção de rota ou também quando uma informação comum estiver sendo transmitida;
- **Hello:** este é o mecanismo de manutenção de rotas utilizado pelo AODV. Periodicamente o nó fonte envia uma mensagem de *Hello* para o nó de destino e aguarda receber uma resposta.

Em geral, as mensagens de RREQ possuem em sua composição o endereço e número de sequência da fonte; endereço e número de sequência do destino; número de saltos e um identificador para a RREQ, que é incrementado a cada nova requisição, recurso que permite diferenciar se uma dada requisição já foi processada antes. Analogamente, as mensagens de resposta RREP possuem o número de identificação da requisição, o número de sequência atual do destino e o endereço da fonte. Além disso, as tabelas de roteamento possuem uma estrutura muito parecida com as apresentadas para o protocolo DSDV, armazenando informações como o endereço de cada destino, próximo salto, número de saltos e número de sequência de cada destino (PERKINS; ROYER, 1999).

Um exemplo de funcionamento do protocolo AODV está ilustrado no diagrama da Figura 17, por meio do qual é possível observar os mecanismos de descoberta e manutenção de rotas, tendo o seu funcionamento exposto no Quadro 4. A descoberta de rotas é utilizada quando um nó fonte necessita enviar dados para um destino e não existe uma rota pré-definida entre eles.

Supondo que é necessário transmitir um pacote de dados do nó sensor fonte A até um nó de destino D, porém não existe um *link* de conexão direta entre ambos, torna-se necessário realizar um encaminhamento do pacote de dados por uma rota até que a informação chegue ao nó de destino. Neste caso, inicialmente o nó A cria uma requisição de rota para D e transmite via *broadcast* para seus vizinhos. Ao receber a requisição de A, o nó B verifica que não é o nó de destino e então encaminha para os seus vizinhos também por *broadcast*, fazendo com que a informação chegue até o nó C e em seguida em D.

Figura 17: Exemplo de funcionamento do protocolo AODV.



Fonte: Adaptado de Parvathi (2012).

Assim que a requisição chegar ao destino (nó D), uma mensagem de resposta RREP será enviada via *unicast* destinada ao nó fonte pela rota reversa, informando o número de sequência atual do destino e a rota identificada. Esta informação é armazenada não só na tabela de roteamento do nó fonte, mas também nos nós pertencentes à rota identificada.

Para assegurar que não ocorram falhas, o protocolo AODV possui um mecanismo de manutenção de rotas bastante simples. Uma mensagem de *Hello* é enviada de acordo com períodos fixos a partir do nó fonte até o destino e este, por sua vez, responde ao nó fonte confirmando a disponibilidade do caminho na rede. Caso ocorra alguma falha durante o encaminhamento desta mensagem, o nó que identificar o problema deverá enviar ao nó fonte uma mensagem de erro (RERR) informando que o caminho não é mais válido,

levando o nó fonte a transmitir uma requisição de rota.

As mensagens de erro não são enviadas apenas durante as manutenções de rota, mas também quando um pacote de dados está sendo transmitido. Este processo também está ilustrado na Figura 17.

Quadro 4: Algoritmo AODV.

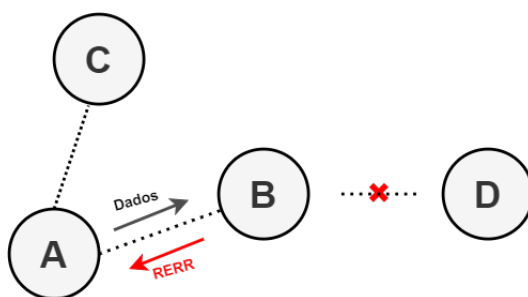
Algoritmo 2: Algoritmo do protocolo AODV	
1	// A = nó fonte, D = nó destino
2	// NS = Número de sequência, TR = Tabela de roteamento
3	if <i>existe uma rota válida para D na TR de A</i> then
4	- O nó A envia os dados para o nó D;
5	else
6	- Um pacote de requisição de rota (RREQ) é criado pelo nó A.
7	- O pacote RREQ é enviado para todos os vizinhos de A.
8	for <i>todos os nós N que receberem a RREQ enviada por A</i> do
9	if <i>a RREQ já foi processada antes</i> then
10	- Descarta RREQ.
11	else
12	if <i>N é o destino (nó D)</i> then
13	- Envia uma mensagem RREP para A.
14	else if <i>N conhece uma rota pra D com NS $\geq$ ao da RREQ</i> then
15	- Envia uma mensagem RREP para A.
16	else
17	- Salva de qual nó recebeu a mensagem RREQ.
18	- Encaminha a RREQ para todos os nós vizinhos.
19	end
20	end
21	while <i>(um nó N receber uma RREP) e (N $\neq$ A)</i> do
22	- Encaminha a RREP pelo caminho de volta.
23	- Salva as informações sobre o nó que está enviando a RREP na TR.
24	end
25	- O nó A recebe a RREP e atualiza sua TR de acordo com as informações recebidas.
26	- A envia dados para o nó D utilizando a nova rota.
27	end
28	end

Fonte: Adaptado de (BOUKERCHE, 2009).

também salvará as suas informações.

Por outro lado, diferente dos protocolos que possuem mecanismos de manutenção de rotas periódicas, no protocolo DSR não há intervalos fixos para realizar esta tarefa. O monitoramento de caminhos válidos é feito durante a própria transmissão de mensagens entre origem e destino (JOHNSON; MALTZ, 1996). Um exemplo deste processo está apresentado na Figura 19, em que o nó A envia ao nó B um pacote de dados destinado ao nó D; entretanto, ao perceber um erro de roteamento, B informa ao nó A por meio de uma mensagem RERR.

Figura 19: Exemplo de notificação de indisponibilidade de rota do protocolo DSR.



Fonte: Elaborado pelo autor.

5.2.3 Híbridos

Os protocolos de roteamento híbridos são nomeados desta maneira pois possuem, como característica, a combinação de recursos dos protocolos pró-ativos e reativos, possibilitando uma utilização conveniente das vantagens de ambos (FARIAS, 2008).

Nesse contexto, levando em consideração que o seu funcionamento está embasado na combinação de recursos de outras categorias de protocolos de roteamento, geralmente é utilizado um roteamento estático nas margens da rede, ou seja, nas extremidades, onde as mudanças de topologia não são tão frequentes. Por outro lado, nas localizações da rede mais propensas a mudanças é utilizada uma abordagem de roteamento dinâmico, sob demanda. Dessa forma, o desempenho geral dos protocolos híbridos mostra-se mais eficiente (BOUKERCHE, 2009), porém, foi decidido explorar neste trabalho apenas os protocolos mais tradicionais (pró-ativos e reativos) deixando a avaliação de protocolos híbridos para trabalho futuros. Um exemplo de protocolo de roteamento híbrido é o *Zone Routing Protocol* (ZRP).

6 Metodologia

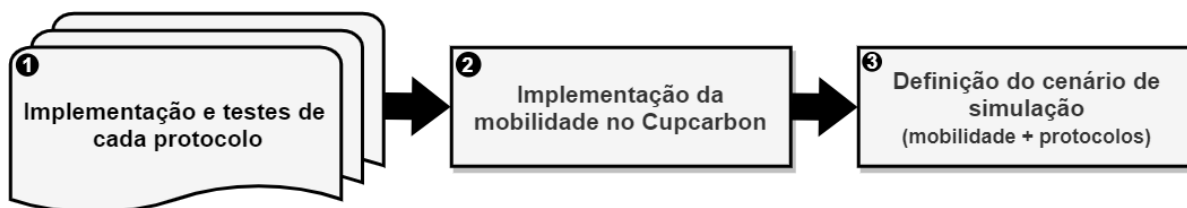
Neste capítulo será apresentada a estrutura de implementação de alguns protocolos de roteamento para redes sem fio, mostrando a topologia de rede utilizada e o cenário de simulação. Além disso, também serão descritas as métricas de avaliação utilizadas para análise do desempenho dos protocolos, a descrição do desenvolvimento de uma ferramenta para auxiliar no cálculo do raio de alcance dos nós, a proposta de um algoritmo de ajuste de potência de transmissão e o detalhamento da montagem experimental realizada.

6.1 Descrição e Estrutura de Implementação dos Protocolos

As simulações desenvolvidas neste trabalho foram realizadas no *software* simulador de redes *Cupcarbon*, tendo sido implementados os protocolos de roteamento DVR, AODV e DSR, englobando categorias de protocolos com características diferentes entre si. Com a finalidade de comparar o desempenho entre tais protocolos em um cenário real, foi considerada uma situação em que um nó sensor fonte está representando um ônibus que desloca-se por ruas e avenidas de um ambiente urbano e está realizando medições da qualidade do ar³, as quais são enviadas para um nó central.

Dessa forma, as implementações neste trabalho foram desenvolvidas em três etapas, ilustradas na Figura 20.

Figura 20: Diagrama com a estrutura de implementação dos protocolos de roteamento.



Fonte: Elaborado pelo autor.

Inicialmente, os protocolos foram simulados desconsiderando eventuais erros que pudessem ocorrer nos nós da rede ou situações de mobilidade e, assim, foram realizados testes para cada protocolo com o propósito de comprovar os passos de roteamento e seu

³Projeto de pesquisa em desenvolvimento pelo grupo de Microengenharia da UFPB, do qual este trabalho faz parte.

funcionamento de maneira geral, comparando com os algoritmos apresentados no capítulo anterior. Em seguida, foi desenvolvido um algoritmo para permitir o deslocamento do nó fonte a partir de coordenadas geográficas específicas e, por fim, foi realizada a integração entre os códigos dos protocolos e o algoritmo de mobilidade.

6.1.1 Organização dos Códigos

A linguagem de programação Senscript está inserida no paradigma de linguagem de *script* e, originalmente, não permite criar funções nos códigos desenvolvidos pelos usuários; para tal, é necessário criar as funções no código-fonte do *Cupcarbon*, implementadas na linguagem Java, para posteriormente serem utilizadas em Senscript. Assim, a implementação dos protocolos de roteamento foi desenvolvida baseada em quatro arquivos de *script* com finalidades que serão detalhadas a seguir.

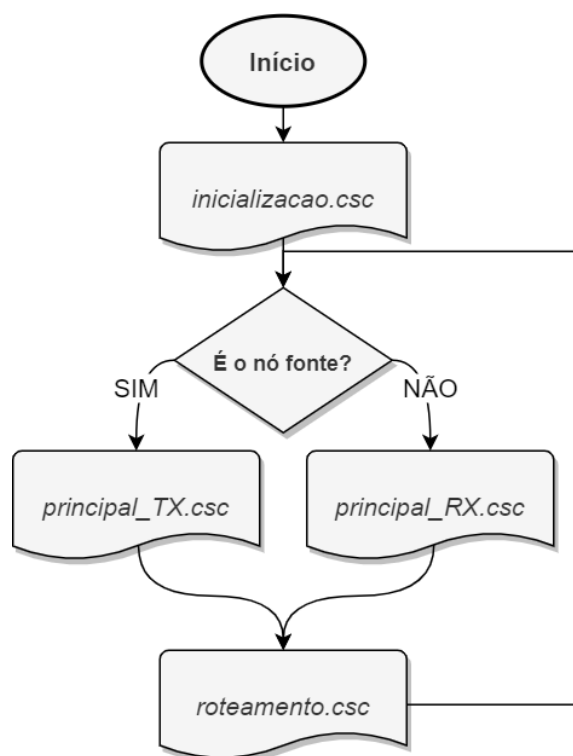
Embora os algoritmos de roteamento de cada protocolo possuam diferenças, nesta primeira etapa foi adotada uma estrutura de organização dos códigos visando manter uma padronização, conforme exposto no fluxograma da Figura 21. Desta forma, de acordo com essa estrutura, cada nó executa inicialmente o *script* nomeado de "*inicializacao.csc*", segundo o qual são criadas as tabelas de roteamento, tabelas de *cache* e também inicializadas as variáveis. Esse *script* também é responsável por guardar as informações das trajetórias (coordenadas geográficas) utilizadas na função de mobilidade.

Após finalizar a execução do código de inicialização, é verificado se o nó é um nó fonte, habilitado para enviar dados ao *gateway*, ou um nó roteador. Em seguida o código principal é executado de acordo com este resultado, sendo o nó fonte associado ao código "*principal_TX.csc*" e os demais ao "*principal_RX.csc*".

No código principal estão contidas as funções de análise das mensagens recebidas, possibilitando ao nó extrair as variáveis do pacote de dados ou simplesmente encaminhar a mensagem. Quando se tratar de um nó fonte, o código principal também possui uma função para envio periódico de dados para o *gateway*. Além disso, também estão incluídas nesses *scripts* as funções de anúncio de rotas para o caso dos protocolos reativos e compartilhamento de tabelas de roteamento para os protocolos pró-ativos.

Caso algum nó da rede identifique que uma informação relativa às rotas foi recebida, é iniciada a execução do *script* "*roteamento.csc*", em que estão incluídos os algoritmos de

Figura 21: Diagrama de organização dos códigos.



Fonte: Elaborado pelo autor.

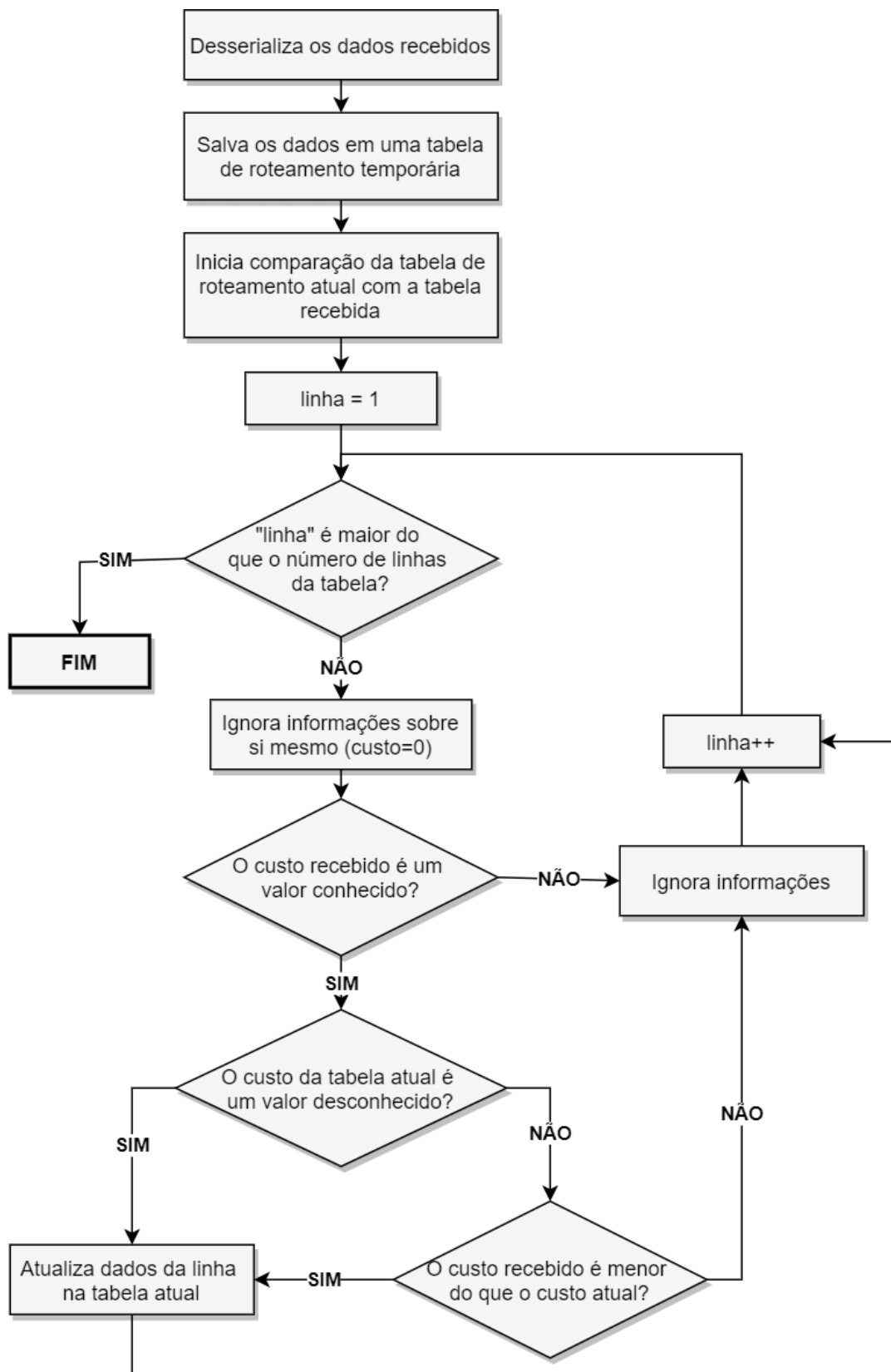
roteamento de cada protocolo. Todas as verificações de métricas, números de sequência e identificação de requisições de rotas são realizadas neste código, retornando sempre para execução dos códigos principais ao finalizar o processo de roteamento.

6.1.2 Implementação dos Protocolos

Nas simulações desenvolvidas neste trabalho, as tabelas de roteamento e as requisições de rota são compartilhadas entre os nós da rede por meio de mensagens serializadas, ou seja, essas mensagens contêm diversas informações agrupadas e delimitadas por um separador "&". Ao receber uma mensagem, os nós reconstroem a informação separando cada dado enviado pelo nó de origem. Sabendo disso, são apresentados agora com maiores detalhes os algoritmos de cada protocolo implementados no *Cupcarbon* a partir do ponto em que uma mensagem de roteamento foi recebida.

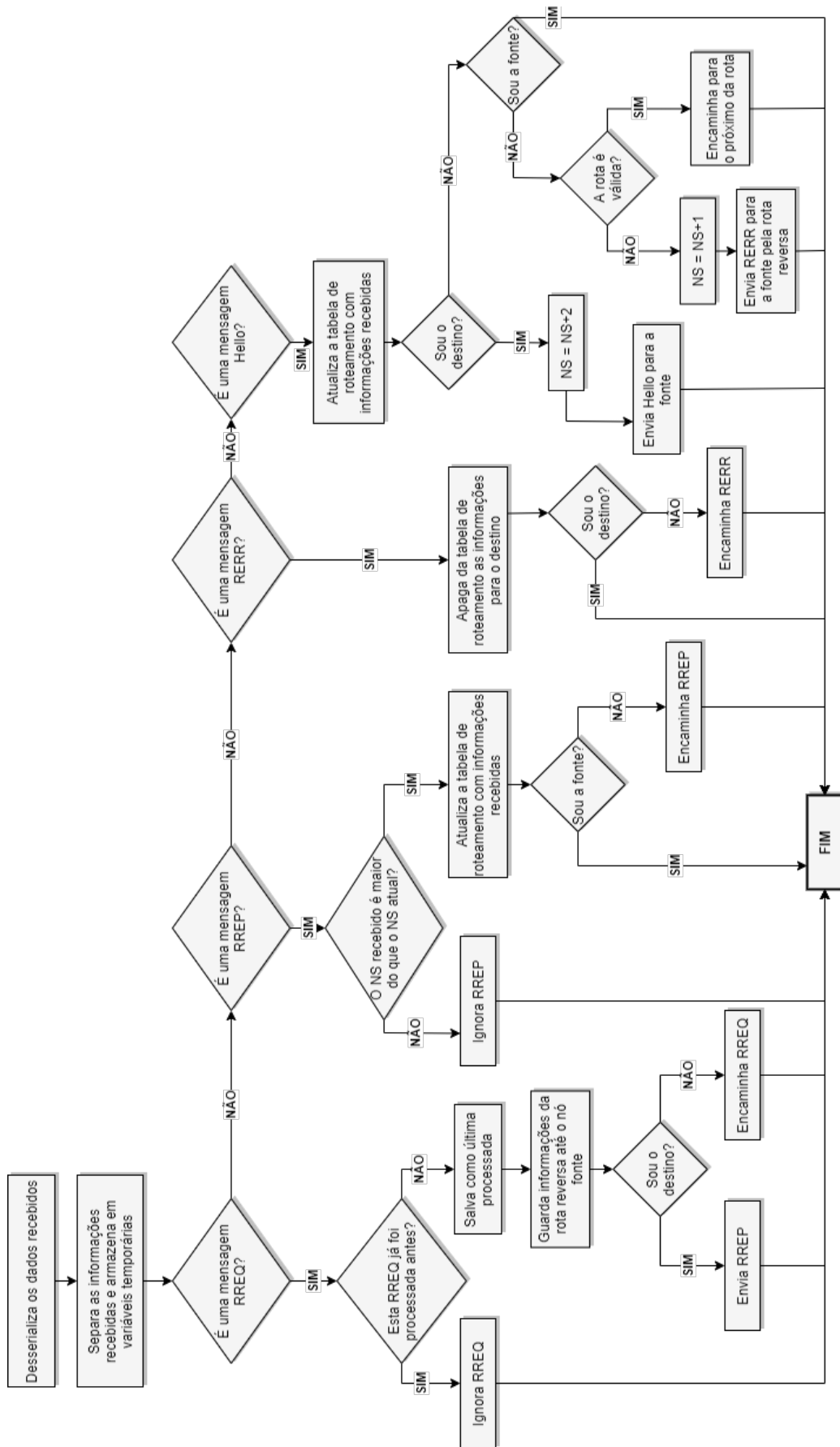
Os fluxogramas apresentados nas Figuras 22, 23 e 24 representam as instruções de roteamento para os protocolos DVR, AODV e DSR que estão implementadas em cada nó da rede e foram desenvolvidos baseados nos exemplos ilustrados no Capítulo 5.

Figura 22: Fluxograma de implementação do roteamento do protocolo DVR.



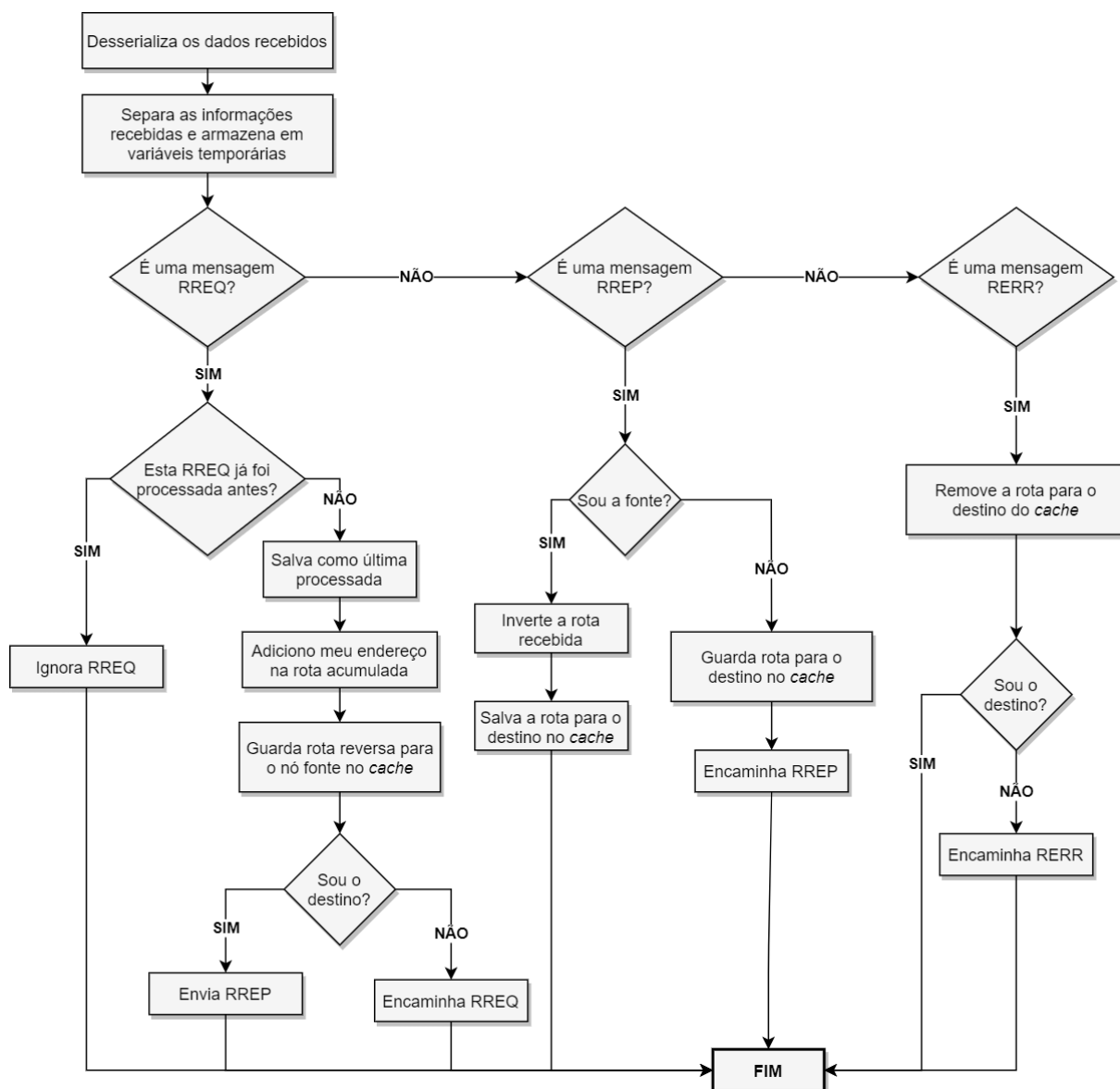
Fonte: Elaborado pelo autor.

Figura 23: Fluxograma de implementação do roteamento do protocolo AODV.



Fonte: Elaborado pelo autor.

Figura 24: Fluxograma de implementação do roteamento do protocolo DSR.

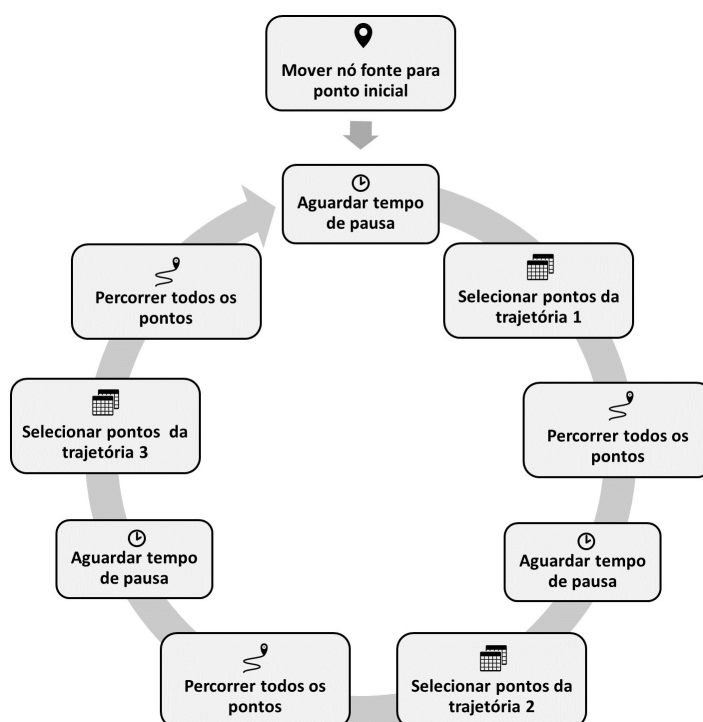


Fonte: Elaborado pelo autor.

6.1.3 Implementação da Mobilidade

Após finalizar as etapas de implementação e validação dos protocolos de roteamento, foi iniciada a segunda etapa de desenvolvimento deste trabalho, que teve como objetivo possibilitar o deslocamento de um nó da rede a partir dados de entrada de latitude e longitude. O simulador *Cupcarbon* possui uma função destinada à movimentação, levando em consideração parâmetros de entrada como coordenadas geográficas (latitude e longitude) e velocidade em metros por segundo. Desta forma, visando a definição do cenário de simulação, que leva em consideração a movimentação do nó fonte, foram desenvolvidas algumas funções para permitir a movimentação de acordo com trajetórias pré-definidas compostas por múltiplas coordenadas geográficas, como mostrado na Figura 25.

Figura 25: Fluxograma de mobilidade.



Fonte: Elaborado pelo autor.

Para garantir que o ponto de partida do nó fonte seja sempre o mesmo, inicialmente este é movido para o primeiro ponto de forma instantânea, ficando parado nesta posição por um tempo que foi chamado de tempo de pausa. Em seguida, a função de mobilidade entra em um ciclo repetitivo, composto basicamente pela seleção dos pontos de cada trajetória nas tabelas armazenadas na memória. A movimentação entre cada ponto selecionado é

executada de acordo com a sintaxe da função em Senscript exibida a seguir, referente à função nativa do *Cupcarbon* para movimentação:

move latitude longitude altitude velocidade

Dessa forma, essas funções desenvolvidas para mobilidade nas simulações foram adaptadas dentro do código "*principal_TX.csc*", que está associado ao nó fonte.

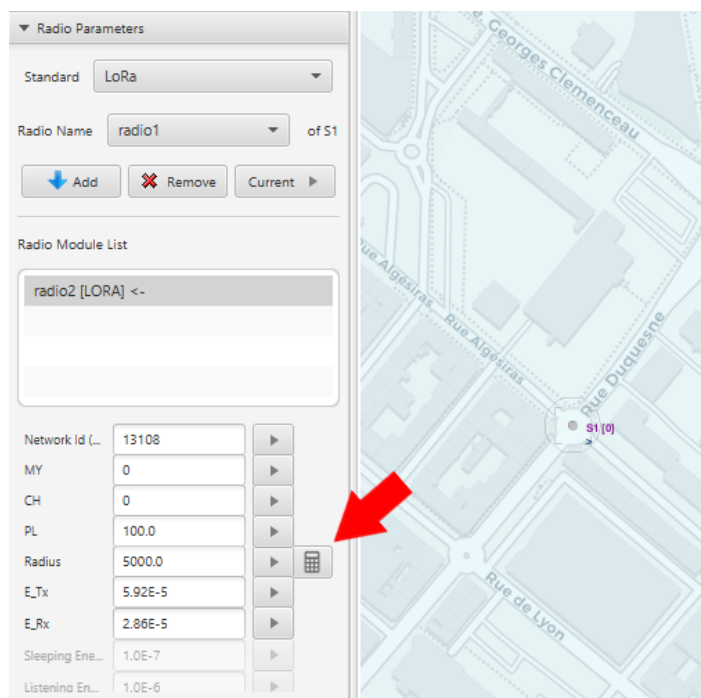
6.2 Desenvolvimento de Ferramenta para Cálculo do Raio de Alcance

Ao selecionar a tecnologia de comunicação sem fio que será utilizada por um nó, o *Cupcarbon* fornece valores padrão para o raio de alcance, sendo estes valores iguais a 100 m, 400 m e 5 km para Zigbee, WiFi e LoRa, respectivamente. Diante disso, um dos objetivos deste trabalho de Dissertação foi desenvolver, dentro da interface do simulador, uma ferramenta que permitisse calcular o raio de alcance baseado no modelo de perda de propagação de Egli e também considerando os parâmetros da modulação LoRa.

Este modelo foi escolhido por ter um equacionamento simples, adequado para a faixa de frequências dos módulos de comunicação LoRa utilizados neste trabalho e que também é apropriado para o raio de distância de cobertura da área na qual serão realizadas as simulações. Além disso, também foi levado em consideração o fato do modelo de Egli ser adequado para situações em que um dispositivo possui mobilidade e o outro permanece fixo. Por outro lado, existem algumas limitações, como por exemplo, o modelo de Egli considera uma linha de visada direta e fornece a perda de propagação média total, não distinguindo entre perda de espaço livre e perdas por outros fatores.

O desenvolvimento desta ferramenta consistiu em uma etapa desta Dissertação e foi realizado utilizando a linguagem de programação Java e integrado ao código-fonte do *Cupcarbon*. Para abrir a interface gráfica é necessário selecionar um nó que esteja utilizando a modulação LoRa, caso contrário o *Cupcarbon* exibe uma mensagem informando da limitação. Ao selecionar o nó para o qual se deseja realizar os cálculos, basta clicar no ícone de calculadora localizado ao lado do campo de raio de alcance da aba de parâmetros do rádio, conforme indicado na Figura 26.

Figura 26: Integração da ferramenta auxiliar com a interface gráfica do *Cupcarbon*.



Fonte: Elaborado pelo autor.

Ao clicar no ícone da calculadora será aberta uma janela, a qual está ilustrada na Figura 27. No topo dessa janela é exibido qual nó foi selecionado pelo usuário e também qual é o seu raio atual em metros. Os parâmetros relacionados à transmissão (SF, frequência e largura de banda) são mostrados logo abaixo, sendo possível alterar o valor do parâmetro SF diretamente nessa interface, o que impacta no resultado do cálculo do raio.

O usuário pode selecionar o módulo de comunicação LoRa que deseja, tendo sido inseridos inicialmente os módulos SX1276, SX1277, SX1278 e SX1279. A seleção do módulo de rádio utilizado afeta o parâmetro de sensibilidade, e as informações utilizadas para este parâmetro foram obtidas no *datasheet* da Semtech (2015), levando em consideração os parâmetros LoRa.

Outros parâmetros de entrada do modelo de Egli também podem ser alterados nessa interface, tais como os ganhos das antenas do receptor e do transmissor, as suas respectivas alturas, assim como a potência de transmissão e, se necessário, uma margem de ruído. Depois de selecionar e alterar todos os valores, o usuário pode clicar no botão "Calcular raio" para obter o resultado calculado e também pode clicar em "Aplicar no mapa" para que este valor calculado seja aplicado ao nó que foi selecionado no mapa de simulação.

Figura 27: Interface gráfica da ferramenta auxiliar para cálculo do raio de alcance.

Modelos de propagação

Modelo de propagação de Egli adaptado para LoRa

Douglas de Farias Medeiros - Laboratório de Microengenharia - UFPB

Nó selecionado = 5 Raio atual (m) = 3000.0

SF = 7 Mudar Frequência = 9.15E8 BW = 125000.0

Rádio LoRa utilizado: SX1276

Ganho da antena do nó selecionado (dB): 3

Ganho da antena do gateway (dB): 3

Altura do nó selecionado (m): 2.5

Altura do gateway (m): 2.5

Margem de ruído (dB): 10

Potência de transmissão (dBm): 20

Sensibilidade do gateway (dBm): -123

Raio calculado (m) = 1560.484492977 Calcular raio Aplicar no mapa

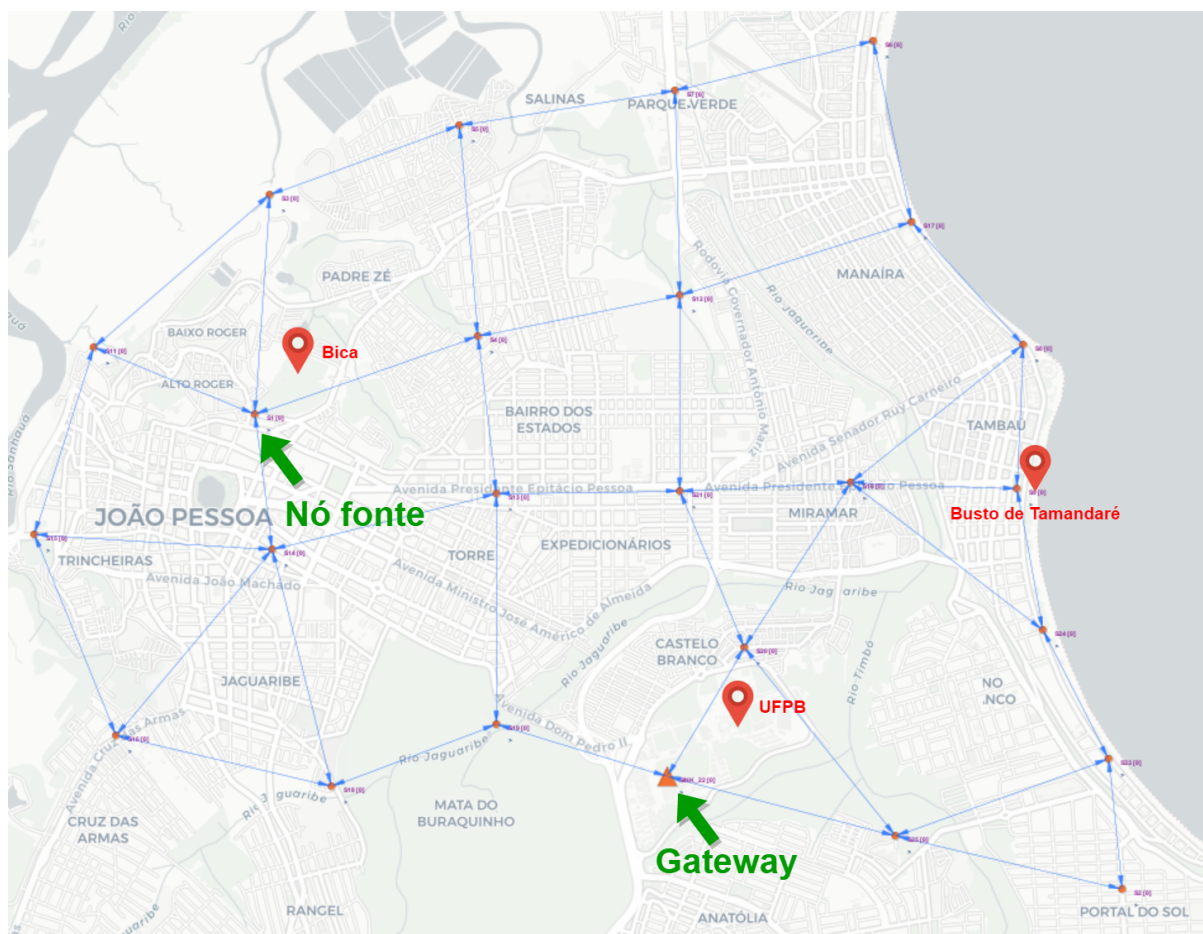
Fonte: Elaborado pelo autor.

6.3 Cenário de Simulação e Topologia de Rede

Para avaliar o desempenho dos protocolos de roteamento abordados neste trabalho de Dissertação, foi escolhido um cenário de simulação que englobasse uma situação de aplicação real inserida no contexto de Cidades Inteligentes. Assim, foi considerada uma situação em que um nó sensor desloca-se por ruas e avenidas de um ambiente urbano, realiza medições de gases poluentes na atmosfera e envia periodicamente esses dados para um *gateway* de localização fixa - o qual poderá encaminhar as medições para um servidor na nuvem, por exemplo, e disponibilizar os dados para os usuários finais por meio de aplicações tal qual *dashboards* em páginas na Internet, aplicativos para celular, gerar alertas de acordo com os níveis de poluição, etc.

Para este cenário de simulação, foi considerada a topologia de rede *mesh*, como mostrada na Figura 28, com uma quantidade total de nós igual a 25, sendo cada nó equipado com um módulo de comunicação sem fio LoRa SX1276 na faixa de frequência de 915 MHz. O raio de alcance dos nós foi considerado aproximadamente 1,56 km e foi calculado

Figura 28: Topologia de rede utilizada nas simulações.



Fonte: Elaborado pelo autor.

utilizado a ferramenta apresentada na Subseção 6.2 para antenas com ganhos de 3 dBi, altura das antenas de 2,5 metros e potência de transmissão de 20 dBm. Além disso, também foram considerados os parâmetros LoRa que estão expostos na Tabela 2. Para cada simulação, também foi considerado um intervalo de tempo total de uma hora, conforme detalhado na Tabela 2.

Tendo em vista que o protocolo DVR é um protocolo pró-ativo, foi definido um período para compartilhamento das tabelas de roteamento igual a 5 minutos, sendo o primeiro compartilhamento iniciado assim que a simulação começa. No caso das simulações do protocolo AODV, foi definido um período para o mecanismo de manutenção de rotas (envio de mensagens *Hello*) igual a 1,5 minutos. A diferença entre os períodos de tempo definidos para os protocolos AODV e DVR justifica-se pelo fato da dinâmica de roteamento ser distinta entre ambos, pois, para o caso do protocolo DVR, o processo é muito mais

Tabela 2: Parâmetros gerais utilizados nas simulações.

Tempo total de simulação	3600 segundos (1 hora)
Tempo de pausa	90 segundos
Raio de alcance de cada nó	1,56 km
Modelo de propagação de rádio	Egli
Quantidade total de nós	25
Tecnologia de comunicação sem fio	LoRa
Frequência de transmissão	915 MHz
<i>Spreading Factor (SF)</i>	7
<i>Code Rate (CR)</i>	4/5
Largura de banda (BW)	125 kHz

Fonte: Elaborado pelo autor.

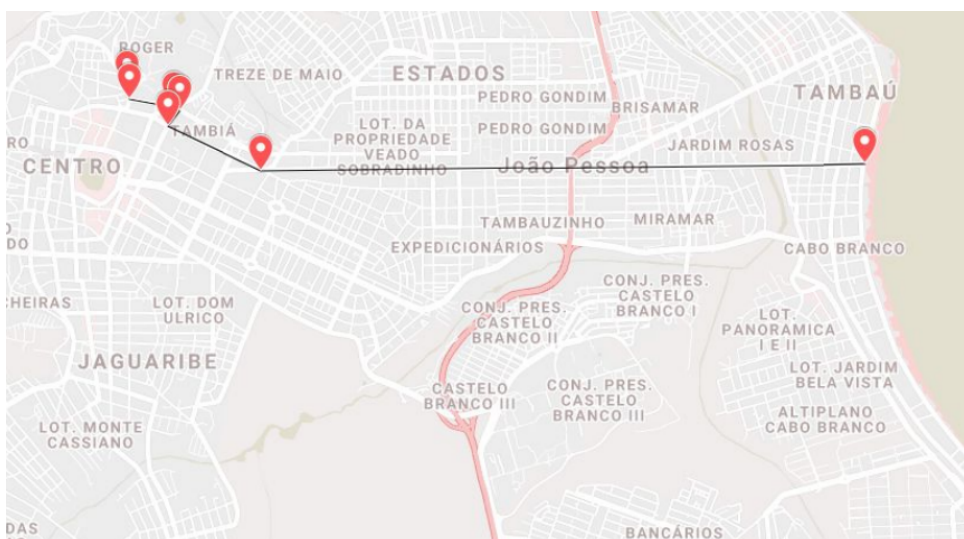
complexo e custoso. E, por fim, foi definido um período de 30 segundos para envio de mensagens gerais, utilizadas para transmissão de pacotes de dados provenientes de sensores.

Diante do fato do nó sensor fonte estar simbolizando um ônibus que desloca-se por ruas de um ambiente urbano, foi escolhida a cidade de João Pessoa-PB como o local de simulação e a velocidade do nó fonte foi variada entre 9 e 180 km/h, enquanto que os outros 24 nós permaneceram estáticos. Assim, foram selecionados três destinos que estão listados a seguir e também indicados na Figura 28. A partir destes destinos foram definidos os trajetos ilustrados na Figura 29:

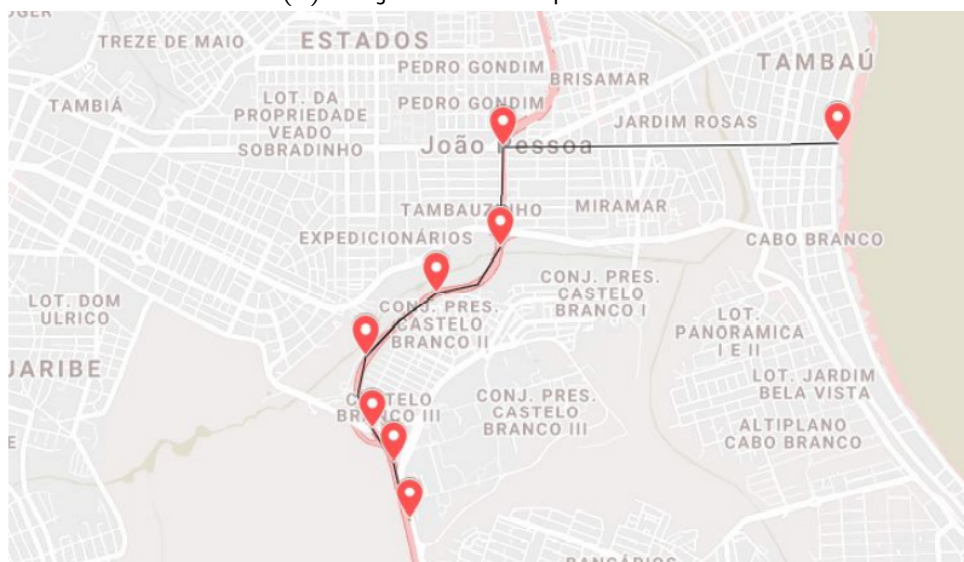
- Parque Zoobotânico Arruda Câmara (Bica);
- Busto de Tamandaré; e
- Campus I da Universidade Federal da Paraíba (UFPB).

O primeiro trajeto compreende o trecho entre a Bica e o Busto de Tamandaré, localizado no bairro de Tambaú, sendo este trajeto composto por 7 pontos. O segundo trajeto, por sua vez, direciona o nó sensor móvel do Busto de Tamandaré para a Universidade Federal da Paraíba por meio de 8 localizações geográficas. O último trajeto que compõe o ciclo percorrido pelo nó fonte o leva da UFPB de volta até a Bica, reiniciando o processo de mobilidade.

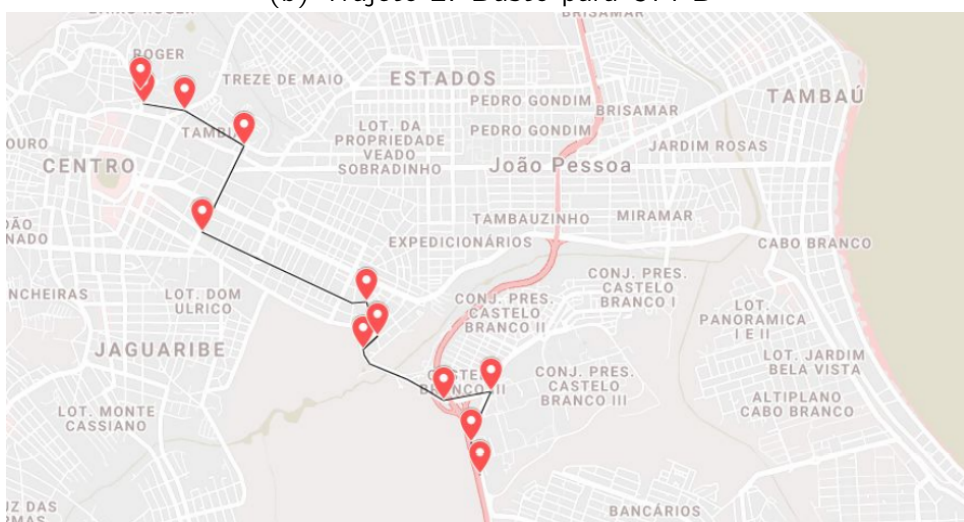
Figura 29: Trajetórias utilizadas no cenário de simulação.



(a) Trajeto 1: Bica para Busto



(b) Trajeto 2: Busto para UFPB



(c) Trajeto 3: UFPB para Bica

Fonte: Elaborado pelo autor.

Ao finalizar um trajeto, quando o nó fonte chega a um ponto de destino, deve ser aguardado um certo tempo antes de iniciar o próximo caminho, sendo este tempo chamado de tempo de pausa (TP). Nas simulações deste trabalho, foi considerado um tempo de pausa igual a 1 minuto.

6.4 Métricas de Avaliação de Desempenho

Para que seja possível atingir alguns dos objetivos deste trabalho, é realizada uma análise de desempenho entre os três protocolos que foram implementados, verificando o seu funcionamento quando submetido à situação descrita no tópico anterior. Para tal, foram adotadas as seguintes métricas: razão de entrega de pacotes (*Packet Delivery Ratio* - PDR), atraso médio fim-a-fim (*Average End-to-end Delay* - E2ED), *Jitter* médio e taxa de transferência (*Throughput* - THR). Além do mais, também foram obtidos os perfis de consumo por nó e por protocolo de acordo com o modelo clássico disponibilizado pelo simulador *Cupcarbon*.

Com o propósito de gerar uma análise de dados mais robusta, com maior rigor estatístico, foram consideradas 5 rodadas de simulação para cada uma das 11 velocidades consideradas, para cada métrica e também para cada protocolo, totalizando uma quantidade de 220 simulações. Ao final, foram obtidos os valores médios para cada métrica, assim como o desvio padrão e o intervalo de confiança (IC) para um nível de confiança de 95% a partir de uma distribuição normal, calculada com auxílio da expressão a seguir (BECKER, 2015):

$$IC = \bar{x} \pm z_{\frac{\alpha}{2}} \left(\frac{\sigma}{\sqrt{n}} \right), \quad (17)$$

sendo $z_{\frac{\alpha}{2}}$ o coeficiente de confiança, α o nível de confiança, $\bar{x}$ a média, σ o desvio padrão e n o tamanho da amostra.

6.4.1 Razão de Entrega de Pacotes

A primeira métrica calculada foi a razão de entrega de pacotes, por meio da qual é possível avaliar quantos pacotes de dados foram entregues com sucesso ao destino. Assim, essa métrica é uma proporção entre o número de pacotes enviados por um nó fonte e o

número de pacotes recebidos pelo destino, sendo calculada da seguinte forma (AL-DHIEF *et al.*, 2018):

$$PDR = \frac{\sum N_{RX_i}}{\sum N_{TX_i}} \times 100\%, \quad (18)$$

em que N_{RX_i} representa o número de pacotes de dados recebidos com sucesso por um nó de destino i e N_{TX_i} é o número de pacotes enviados para este dispositivo.

6.4.2 Atraso Médio Fim-a-Fim

O atraso médio fim-a-fim é definido como o tempo médio gasto para que uma quantidade de pacotes de dados sejam transmitidos com sucesso pela rede a partir de um nó fonte até um destino específico (AL-DHIEF *et al.*, 2018). Este tempo de atraso é composto pelas componentes de tempo de espera na fila de transmissão, tempo de propagação, tempo de transmissão e processamento (NETO, 2011). Matematicamente, pode ser definido por:

$$E2ED = \frac{\sum T_{RX_i} - T_{TX_i}}{N_{RX_i}}, \quad (19)$$

em que, neste caso, N_{RX_i} é o número de pacotes de dados recebidos pelo nó i , T_{RX_i} representa o tempo em que o pacote de dados foi recebido e T_{TX_i} o tempo em que o pacote foi enviado.

6.4.3 Jitter Médio

O *jitter* médio representa a variação do atraso médio fim-a-fim na entrega de pacotes de dados em uma rede, podendo ocasionar uma situação de não regularidade na recepção dos pacotes de dados, dado por (EMA *et al.*, 2020; SAMPOORNAM; RAAGA, 2019):

$$Jitter = \frac{\sum E2ED_{i_k} - E2ED_{i_{k-1}}}{N_{RX_i}}, \quad (20)$$

sendo $E2ED_{i_k}$ o atraso médio de um dado instante k , $E2ED_{i_{k-1}}$ o atraso médio do instante anterior e N_{RX_i} o número de pacotes de dados recebidos pelo nó i .

6.4.4 Taxa de Transferência

A taxa de transferência, também chamada de vazão ou *throughput*, refere-se ao número de pacotes de dados que foram transmitidos com sucesso ao seu destino final em um determinado canal de comunicação por unidade de tempo (RAHMAN; HASAN; ISLAM, 2012). Geralmente é expresso em bits por segundo (bps), mas também pode ser calculado em termos de pacotes por segundo. Pode ser calculado como:

$$THR = \frac{N_{RX_i}}{\text{Tempo total de simulação}}. \quad (21)$$

6.4.5 Modelos de Consumo de Energia

O simulador *Cupcarbon* possibilita estimar o consumo de bateria de cada nó da rede utilizando dois modelos de consumo chamados de modelo clássico e modelo de Heinzelman. Logo, para visualizar não só o desempenho geral dos protocolos de roteamento, mas também quanto cada um consome em média em determinada janela de tempo, foram levantadas essas informações para todas as simulações considerando o modelo clássico.

Em ambos os modelos de consumo, o *Cupcarbon* utiliza algumas expressões para calcular o quanto de carga de bateria foi gasto na transmissão e na recepção dos pacotes de dados, adotando alguns parâmetros que estão apresentados no Quadro 5.

Quadro 5: Parâmetros para estimação do consumo de energia.

Parâmetro	Descrição
n	Número de bits transmitidos/recebidos
R	Alcance do rádio durante a transmissão
P	Porcentagem da potência de transmissão (0 a 100%)
T	Temperatura do ambiente
E_{TXb}	Consumo de energia para transmissão de 1 byte
E_{RXb}	Consumo de energia para recepção de 1 byte

Fonte: (VIRTUALYS *et al.*, 2019).

Para realizar o cálculo do consumo de energia de um nó sensor no *Cupcarbon* pelo modelo clássico, é necessário fornecer um valor conhecido para o consumo da transmissão e recepção de 1 byte. Assim, este valor pode ser medido em laboratório e alterado na interface do *Cupcarbon* conforme a aplicação.

Entretanto, caso não seja possível identificar este parâmetro, pode ser utilizado o valor padrão, baseado nos experimentos realizados por Wander *et al.* (2005), segundo os quais foram levantados os valores de consumo para uma aplicação utilizando a plataforma de prototipação *Mica2dot*, que possui um microcontrolador de 8 bits e um módulo de rádio. Logo, a quantidade de energia gasta para transmissão de 1 byte verificada nestes experimentos foi de $59,2 \mu\text{J}$ e para recepção foi de $28,6 \mu\text{J}$.

A partir da escolha do módulo de comunicação LoRa SX1276 para ser utilizado nas simulações e na montagem experimental (detalhada mais adiante), foram adotados os valores de consumo disponíveis no *datasheet* da plataforma de desenvolvimento *Feather M0 RFM95*, produzida pela Adafruit, e que possui um processador ARM Cortex M0 (48 MHz/3,3 V) e ADC de 12 bits de resolução (ADAFRUIT, 2019). Os dados relativos ao consumo para envio de 20 *bytes* em função do modo de operação estão organizados na Tabela 3. Como a tensão de alimentação do módulo é de 3,3 V e o tempo de transmissão/recepção para envio dos 20 *bytes* foi de 70 ms, os valores de consumo de energia calculados para 1 *byte* e utilizados nas simulações deste trabalho foram: $E_{TX_b} = 1501,5 \mu\text{J}$ e $E_{RX_b} = 462 \mu\text{J}$.

Tabela 3: Consumo de corrente do módulo SX1276 de acordo com o modo de operação.

Modo de operação	Consumo de corrente
<i>Sleep</i>	300 μA
Transmissão	130 mA
Recepção	40 mA

Fonte: (ADAFRUIT, 2019).

6.4.5.1 Modelo Clássico

O modelo clássico é o modelo de consumo de energia padrão adotado pelo simulador *Cupcarbon*, que leva em consideração alguns dos parâmetros mostrados no Quadro 5. Para calcular o quanto um nó consumiu de bateria durante uma transmissão de um pacotes de dados, pode ser utilizada a seguinte expressão (VIRTUALYS *et al.*, 2019):

$$E_{TX} = \left(\frac{n}{8}\right) \times E_{TX_b} \times P, \quad (22)$$

em que n é o número de bits transmitidos ou recebidos, E_{TX_b} é o consumo de energia para

transmissão de 1 *byte* e P representa a potência de transmissão em valores percentuais.

De forma similar, para obter o gasto de energia durante a recepção dos dados, pode ser feito conforme a seguir:

$$E_{RX} = \left(\frac{n}{8}\right) \times E_{RXb}, \quad (23)$$

em que n é o número de bits transmitidos ou recebidos e E_{RXb} corresponde ao consumo de energia para recepção de 1 *byte*.

6.4.5.2 Modelo de Heinzelman

O segundo modelo de consumo de energia disponível no *Cupcarbon* é baseado no modelo de um rádio de primeira ordem (HEINZELMAN; CHANDRAKASAN; BALAKRISHNAN, 2000). Segundo este modelo, para transmitir um pacote de dados é preciso levar em consideração a energia consumida pelo transmissor, assim como a energia consumida pelo amplificador, conforme expresso matematicamente a seguir:

$$E_{TX} = (E_{elec} \times n) + (E_{amp} \times n \times [R \times P]), \quad (24)$$

em que E_{elec} representa a energia consumida pelo rádio para transmissão ou recepção de um bit e E_{amp} a energia consumida pelo amplificador, n o número de bits da mensagem, R o raio de alcance do rádio e P a potência de transmissão. Por outro lado, o consumo do receptor leva em conta apenas a energia gasta para recepção de um bit e o número de bits de dados n :

$$E_{RX} = E_{elec} \times n. \quad (25)$$

6.5 Algoritmo de Ajuste de Potência de Transmissão

Após realizar as simulações descritas nos tópicos anteriores, foi elencado um protocolo de roteamento com base no melhor desempenho geral para ser utilizado em conjunto com o algoritmo de ajuste de potência de transmissão proposto neste tópico.

6.5.1 Cálculo do RSSI no *Cupcarbon*

O simulador *Cupcarbon* não possui uma função para obter o RSSI (indicador de intensidade de sinal recebido) da última mensagem recebida; porém, a função DRSSI da linguagem Senscript fornece o valor da distância entre um determinado nó e algum outro que tenha enviado dados para esse nó. Dessa forma, foi realizada uma adaptação no *Cupcarbon* para calcular o RSSI a partir dessa distância utilizando o modelo de log-distância. Este modelo foi escolhido pois independe da frequência de transmissão e dos ganhos das antenas. Além disso, também já foi utilizado para comparação com resultados experimentais por Villarim *et al.* (2019) na cidade de João Pessoa-PB, local de simulação deste trabalho. O modelo de perda de caminho de log-distância pode ser obtido pela expressão a seguir (RAPPAPORT, 2001):

$$\overline{PL}(d) = \overline{PL}(d_0) + 10n \log \left(\frac{d}{d_0} \right), \quad (26)$$

em que n corresponde ao coeficiente de perda de caminho (sendo utilizado o valor de 3,3 nas simulações deste trabalho), d é a distância entre o transmissor e o receptor e d_0 é a distância de referência. A escolha do valor de n foi realizada levando em consideração os resultados experimentais obtidos por Villarim *et al.* (2019). Nestes experimentos foram utilizados dois dispositivos que trocaram dados utilizando a modulação LoRa em ambiente urbano, sendo avaliados os critérios de taxa de entrega de pacotes e RSSI em função da distância de transmissão. Em seguida, foram obtidas e comparadas as curvas de RSSI versus distância a partir dos dados experimentais e também a partir do modelo de log-distância, sendo determinado o valor de n igual a 3,3 para melhor ajuste entre as curvas. O valor de $\overline{PL}(d_0)$ representa a perda de caminho para uma linha de visada direta em relação à distância de referência e pode ser obtido por (RAPPAPORT, 2001):

$$\overline{PL}(d_0) = 20 \log \left(\frac{4\pi d_0}{\lambda} \right). \quad (27)$$

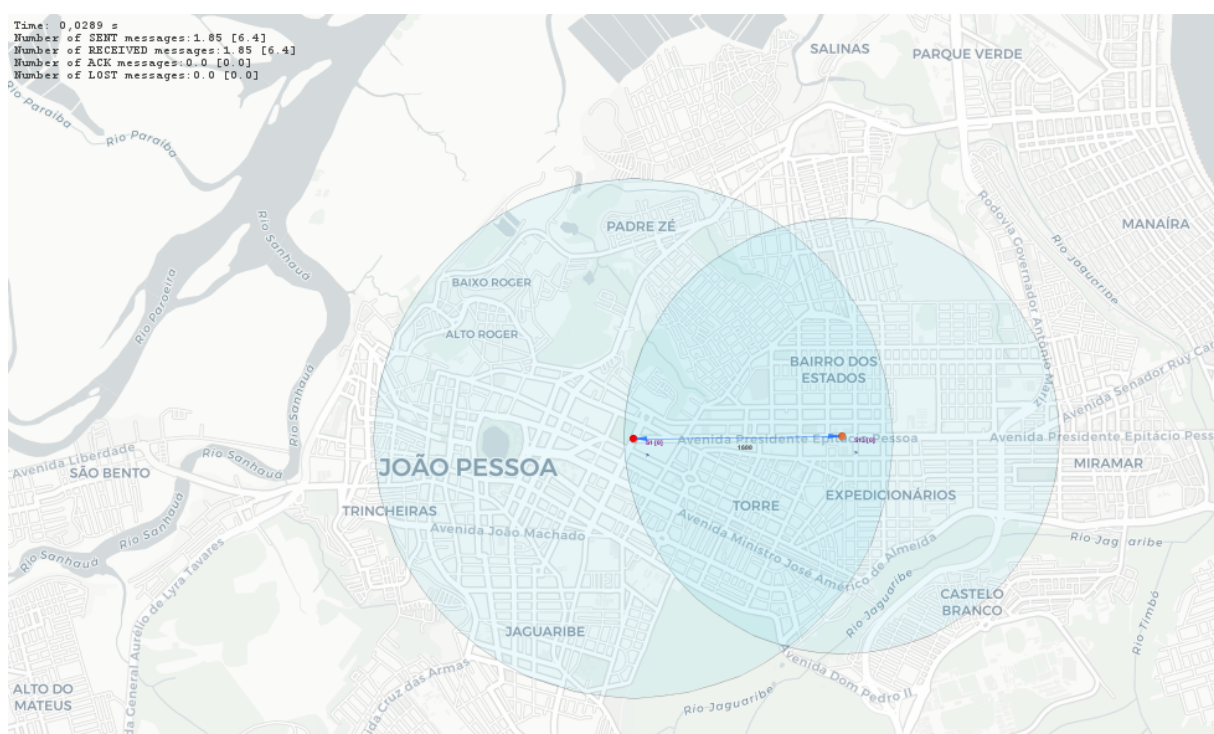
Na Equação 27, d_0 é a distância de referência e o parâmetro λ corresponde ao comprimento de onda da portadora. Assim, as Equações 26 e 27 foram inseridas nos códigos de todos os nós da rede e sempre que uma mensagem é recebida, a distância entre os nós que trocaram dados é obtida por meio da função DRSSI e, em seguida, é calculado o RSSI por meio do modelo de log-distância.

6.5.2 Equação de Potência de Transmissão versus RSSI

A partir da possibilidade de obtenção do RSSI do último pacote de dados recebido utilizando o modelo de log-distância no *Cupcarbon*, foram realizadas simulações variando a distância entre dois nós com o propósito de avaliar a potência de transmissão mínima necessária para que o enlace de comunicação não seja perdido ou prejudicado. A Figura 30 ilustra o cenário dessa simulação.

Dessa maneira, um nó receptor foi mantido fixo e o nó transmissor foi movido inicialmente para uma distância igual ao raio de alcance entre os dispositivos de 1,7 km; além disso, a potência de transmissão na posição inicial do nó móvel foi ajustada para o valor máximo (100% = 20 dBm). Assim, a distância entre os nós foi sendo reduzida em intervalos de 50 metros até que a distância mínima de 1 metro fosse atingida, sendo calculado o valor do RSSI para cada distância desse intervalo. Ao mesmo tempo em que a distância foi reduzida, em cada novo ponto de deslocamento a potência de transmissão também foi reduzida em intervalos de 5% até um valor mínimo em que o enlace de comunicação não fosse perdido.

Figura 30: Simulação realizada para obtenção da expressão de potência de transmissão versus RSSI.

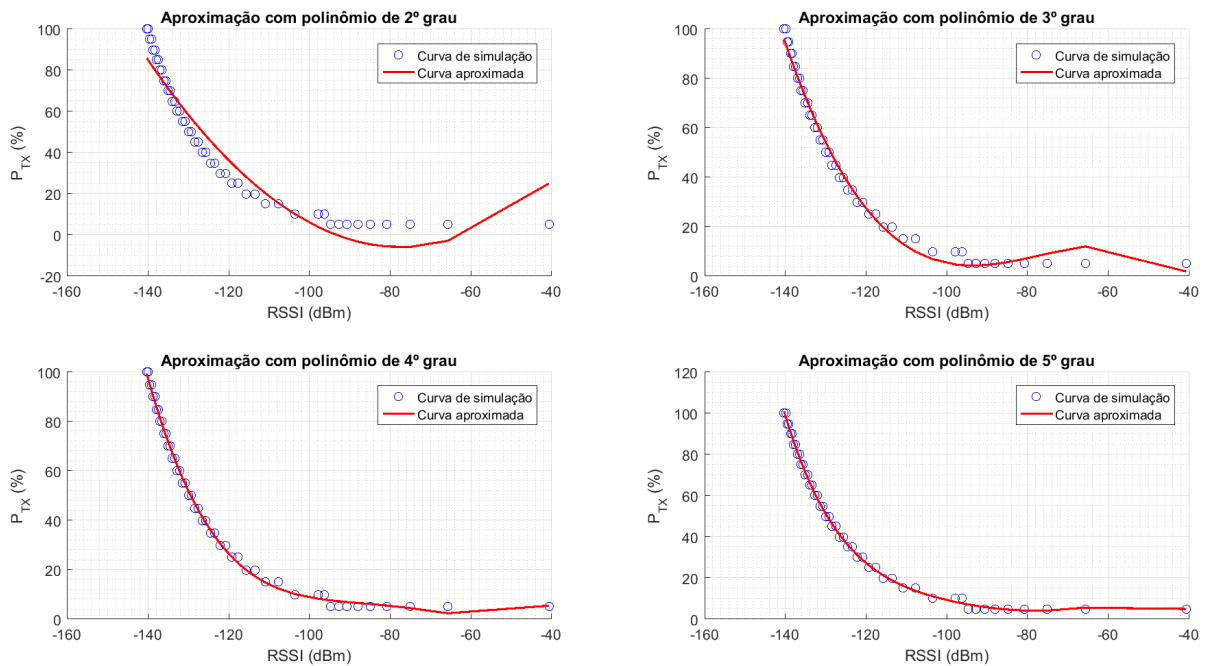


Fonte: Elaborado pelo autor.

Os resultados dessa simulação foram importados para o *software* Octave e, em seguida, foi realizado um processo de interpolação aplicando a função *polyfit*. Na Figura 31 estão exibidos os resultados para as curvas com polinômios de segundo, terceiro, quarto e quinto graus obtidos a partir dos valores de RSSI que foram calculados para cada distância em função da potência ajustada. Para avaliar qual polinômio obteve uma melhor aproximação dos dados originais, foi utilizado como critério a Raiz do Erro Quadrático Médio (do inglês, *Root Mean Square Error* - RMSE). O RMSE pode ser calculado conforme expressão a seguir (AGUIAR, 2018):

$$RMSE = \sqrt{\frac{\sum (X_{real} - X_{simulado})^2}{n}}. \quad (28)$$

Figura 31: Comparação dos polinômios de aproximação dos dados experimentais de potência de transmissão e RSSI.



Fonte: Elaborado pelo autor.

Com base nos valores de RMSE calculados para os quatro polinômios considerados na Figura 31 e expostos na Tabela 4, decidiu-se utilizar o polinômio de quinto grau, pois foi o que mais se aproximou dos resultados de simulação. Esse polinômio, exibido na Equação 29, foi adaptado no código de todos os nós da rede e por meio dele é possível determinar um nível mínimo necessário para a potência de transmissão, de acordo com o

RSSI entre dois nós. É importante destacar que esta equação não pode ser generalizada para quaisquer situações, pois é específica para o cenário de simulação deste trabalho; contudo, o procedimento realizado pode ser replicado com o propósito de obter novas equações para outros cenários.

Tabela 4: RMSE para as curvas de aproximação de potência de transmissão versus RSSI.

Grau do polinômio	RMSE
2º grau	8,1943
3º grau	2,9505
4º grau	1,5623
5º grau	1,2495

Fonte: Elaborado pelo autor.

$$\begin{aligned}
 P_{Tx}(RSSI) = & - 1,161169468381691 \times 10^{-7} \times (RSSI)^5 \\
 & - 4,870470506506291 \times 10^{-5} \times (RSSI)^4 \\
 & - 0,008114743288230 \times (RSSI)^3 \\
 & - 0,660852476932444 \times (RSSI)^2 \\
 & - 25,953469491107786 \times RSSI \\
 & - 3,829454402717133 \times 10^2
 \end{aligned} \tag{29}$$

6.5.3 Fluxograma

Após encontrar uma expressão matemática que fornece o nível mínimo de potência de transmissão em função do RSSI, foi desenvolvido um algoritmo que pode ser adaptado ao código dos protocolos de roteamento considerados neste trabalho. A decisão de qual protocolo foi utilizado em conjunto com o algoritmo teve como base a avaliação de desempenho realizada no capítulo seguinte. Dessa forma, foi escolhido o protocolo DSR para ser utilizado com o algoritmo proposto, pois este protocolo apresentou bom desempenho geral nos resultados para as métricas, assim como o menor consumo de energia, sendo estes resultados discutidos com maiores detalhes no capítulo seguinte.

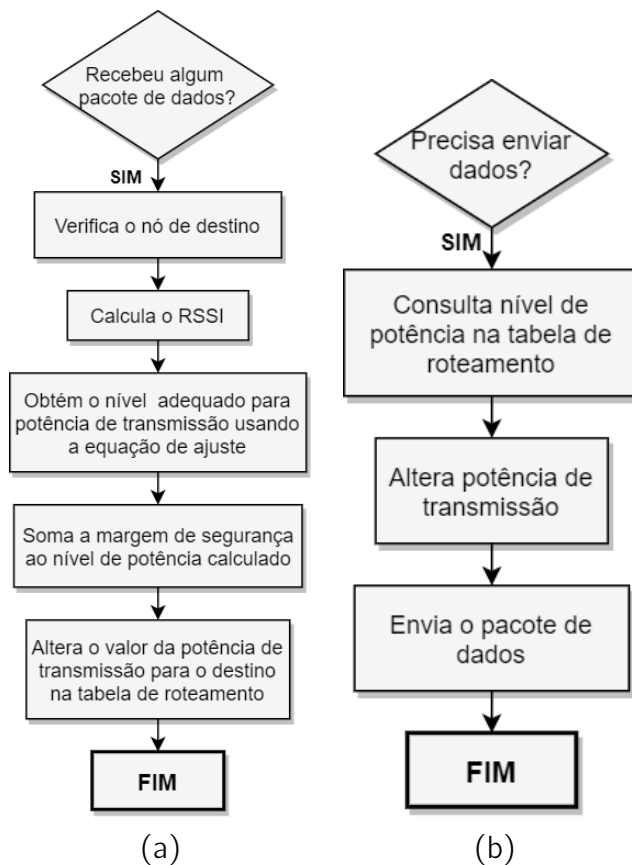
Para utilizar o algoritmo de ajuste de potência proposto, é necessário adicionar uma nova coluna à tabela de roteamento que contém o valor da potência de transmissão para cada destino, sendo este valor inicializado com 100% e alterado conforme o andamento da execução do código. Também foi considerada uma margem de segurança (de 2%) somada

ao valor calculado pela Equação 29, com o objetivo de fornecer uma maior confiabilidade à transmissão dos pacotes de dados.

Na Figura 32.(a) está exemplificado o processo usado para identificar o nível de potência de transmissão mínimo necessário a partir da recepção de um pacote de dados para estabelecer comunicação com um nó de destino. O primeiro passo é identificar qual é o endereço do nó que enviou os dados e em seguida calcular o RSSI. Este valor de RSSI será utilizado como parâmetro de entrada na Equação 29 para obter o valor de potência de transmissão, o qual é somado a um valor de margem de segurança e salvo na coluna da tabela de roteamento que guarda as informações de nível de potência referente àquele nó de destino.

Quando for necessário realizar uma transmissão, o nó fonte consulta na sua tabela de roteamento o valor de nível de potência, realiza o ajuste em seu código e, por fim, envia o pacote de dados. Esse procedimento está ilustrado na Figura 32.(b).

Figura 32: Fluxogramas do ajuste de P_{tx} . (a) Identificação do nível de potência de transmissão de um nó de destino. (b) Ajuste de potência antes de transmitir dados.



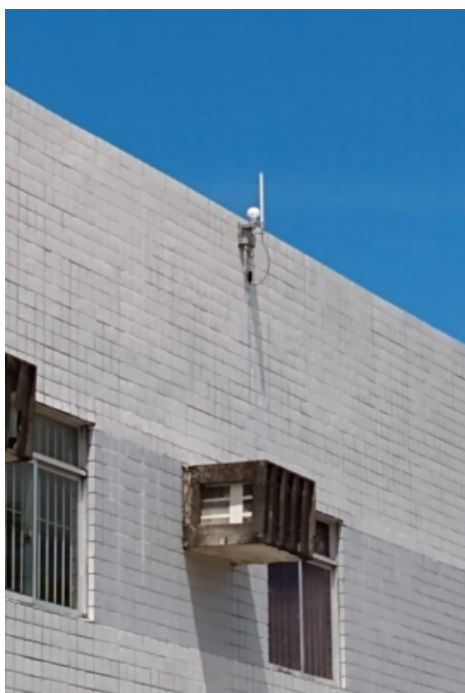
Fonte: Elaborado pelo autor.

6.6 Descrição da Implementação Experimental

O último objetivo deste trabalho foi realizar a implementação experimental do código de algum dos protocolos de roteamento que foram analisados em simulação. A escolha de qual protocolo foi utilizado, assim como no caso do algoritmo de ajuste de potência, foi realizada usando como critério base a avaliação de desempenho do capítulo seguinte, sendo também escolhido o protocolo DSR.

Durante o desenvolvimento da etapa experimental, foi instalado no topo do Laboratório de Microengenharia do Campus I da Universidade Federal da Paraíba um *gateway* industrial do fabricante RisingHF habilitado para o protocolo LoRaWAN, conforme exibido na Figura 33. Este equipamento foi utilizado nesta etapa do trabalho para recepção dos dados enviados por um nó fonte.

Figura 33: Instalação do *gateway* industrial no Campus I da Universidade Federal da Paraíba.



(a)



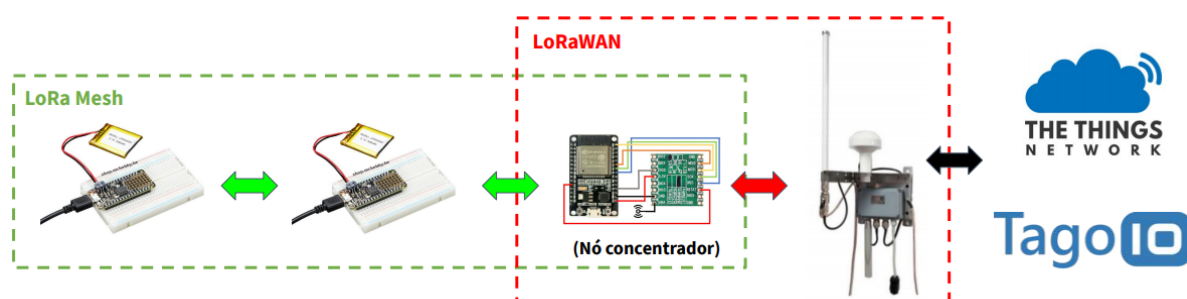
(b)

Fonte: Elaborado pelo autor.

A rede experimental foi dividida em 2 subredes, sendo a primeira uma rede LoRa em malha e a segunda uma rede LoRaWAN em estrela, compondo uma rede com topologia semelhante a uma rede em árvore. Na primeira subrede todos os nós estão habilitados para trocar dados entre si utilizando o protocolo DSR, sendo composta por 1 nó fonte,

1 nó roteador e 1 nó concentrador. O nó concentrador realiza a interface entre as duas subredes da topologia principal e é o responsável por encaminhar os dados recebidos a partir de outros nós para o *gateway*. Por fim, o *gateway* envia os dados para a nuvem, especificamente para o servidor de rede *The Things Network* (TTN). O TTN possui uma integração com a plataforma TagoIO e, sempre que os dados são recebidos, é possível visualizá-los em tempo real através de *dashboards* quem podem ser compostos por componentes tais como indicadores numéricos, gráficos, tabelas, entre outros. A topologia proposta pode ser observada na Figura 34.

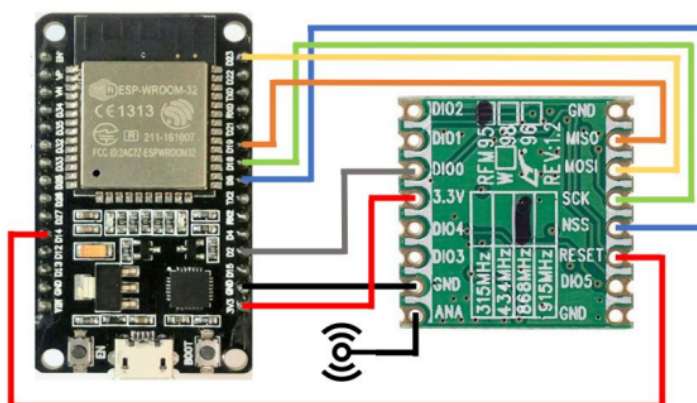
Figura 34: Topologia da montagem experimental.



Fonte: Elaborado pelo autor.

Para montagem dos nós da rede experimental, foram utilizadas duas placas de desenvolvimento Feather M0 RFM95 com rádio LoRa integrado produzida pela Adafruit e uma placa ESP32 em conjunto com o módulo SX1276 para compor o nó concentrador. O esquema de ligação entre o módulo SX1276 e a ESP32 está exibido na Figura 35.

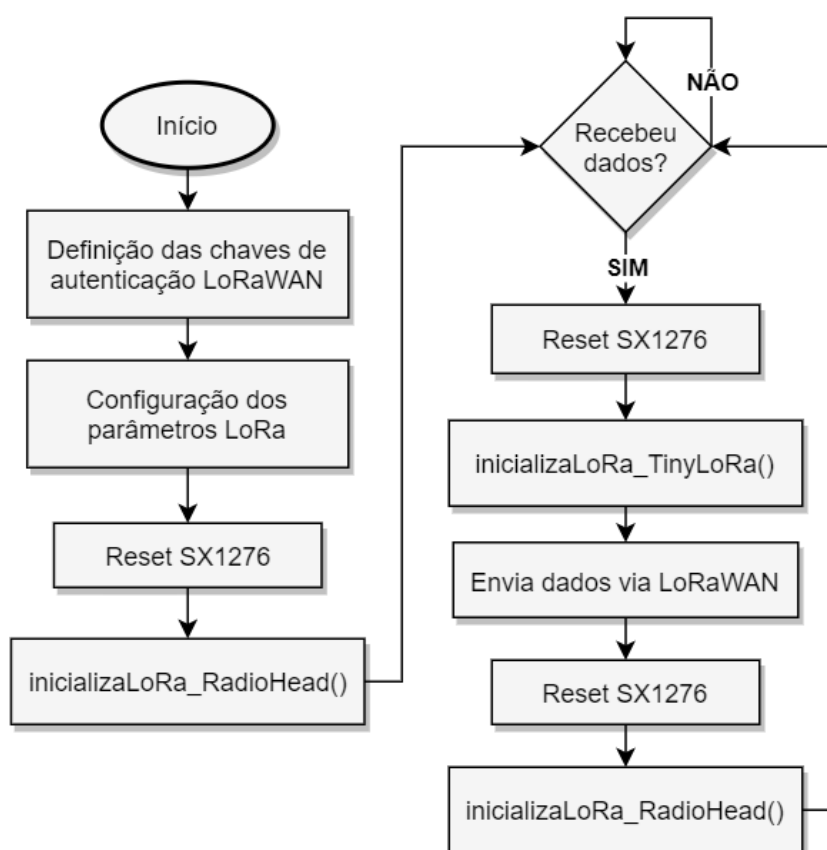
Figura 35: Esquema de ligação entre a placa ESP32 e o módulo de comunicação SX1276.



Fonte: Elaborado pelo autor.

O código da montagem experimental foi desenvolvido a partir de uma adaptação de um exemplo disponibilizado no site da biblioteca *RadioHead* e utilizou duas bibliotecas para comunicação via LoRa com os protocolos DSR e LoRaWAN. Os nós que estão na subrede em malha realizam o roteamento dos dados usando o protocolo DSR via modulação LoRa e usam a biblioteca *RadioHead* (MCCAULEY, 2020b) para esta finalidade. Por outro lado, o nó concentrador utiliza a biblioteca *TinyLoRa* (ADAFRUIT, 2020) para encaminhar os dados para o *gateway* via protocolo LoRaWAN, mas também deve ser capaz de receber dados roteados via modulação LoRa. Sendo assim, o fluxograma exposto na Figura 36 foi implementado no nó concentrador em conjunto com o protocolo do roteamento para habilitá-lo a realizar a interface entre as subredes.

Figura 36: Fluxograma do código de interface do nó concentrador.



Fonte: Elaborado pelo autor.

A implementação experimental do protocolo DSR foi desenvolvida com base no código de roteamento de um exemplo disponibilizado por McCauley (2020a). Esse código de exemplo foi originalmente desenvolvido para ser utilizado com os módulos de comunicação

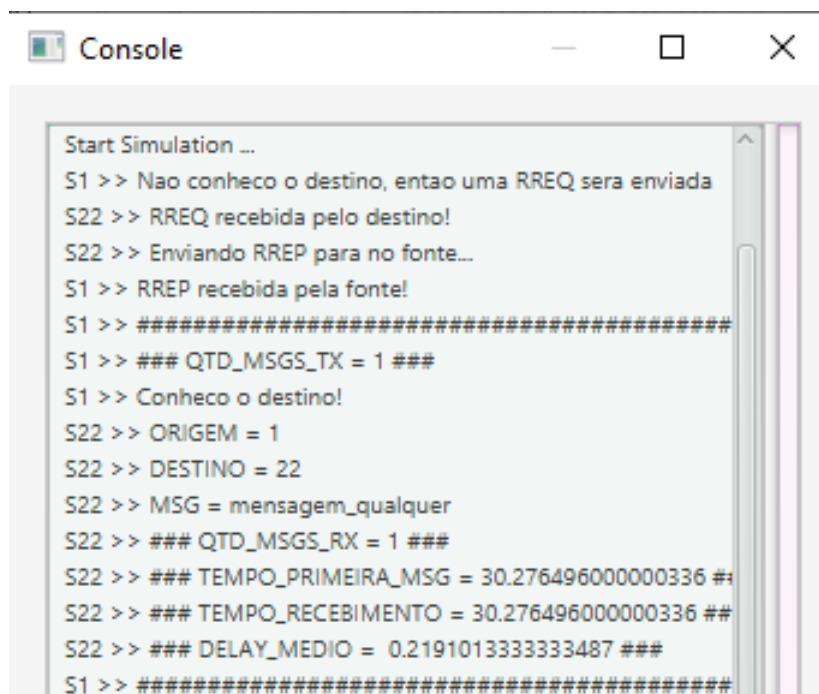
RF22B e RF23B do fabricante HopeRF, os quais utilizam a modulação FSK (do inglês, *Frequency-shift keying*). Assim, foi realizada uma adaptação nos códigos do exemplo visando possibilitar a utilização com os módulos LoRa SX1276. Além do mais, após análise dos códigos, verificou-se que o exemplo já possuía as definições dos tipos de mensagem de roteamento utilizadas pelo protocolo DSR, assim como uma função de descoberta de rotas que estava de acordo com o algoritmo do DSR.

7 Resultados e Discussões

Neste capítulo são apresentados e discutidos os resultados de simulação para os protocolos de roteamento DVR, AODV e DSR quando submetidos ao cenário de simulação em ambiente urbano, visando a análise e avaliação do desempenho destes protocolos. Também será justificada a escolha da margem de segurança que será utilizada para implementação do algoritmo de ajuste de potência de transmissão em conjunto com um dos protocolos abordados neste trabalho e, por fim, serão exibidos os resultados experimentais.

Todas as simulações foram realizadas na versão 4.1 do *Cupcarbon* em uma máquina com processador Intel i5 (2,5 GHz) e 8 GB de memória RAM. As métricas de avaliação foram calculadas pelo *gateway* durante a execução das simulações e exibidas no *console* do simulador, conforme exibido na Figura 37, sendo copiadas para uma planilha do Excel após o tempo total de simulação. Ao finalizar todas as simulações, os dados foram carregados no *software* Octave e, assim, levantadas as curvas para cada métrica.

Figura 37: Exemplo de apresentação do cálculo das métricas no *console* do *Cupcarbon*.



```

Start Simulation ...
S1 >> Nao conheco o destino, entao uma RREQ sera enviada
S22 >> RREQ recebida pelo destino!
S22 >> Enviando RREP para no fonte...
S1 >> RREP recebida pela fonte!
S1 >> #####
S1 >> ### QTD_MSGS_TX = 1 ###
S1 >> Conheco o destino!
S22 >> ORIGEM = 1
S22 >> DESTINO = 22
S22 >> MSG = mensagem_qualquer
S22 >> ### QTD_MSGS_RX = 1 ###
S22 >> ### TEMPO_PRIMEIRA_MSG = 30.276496000000336 #
S22 >> ### TEMPO_RECEBIMENTO = 30.276496000000336 ##
S22 >> ### DELAY_MEDIO = 0.2191013333333487 ###
S1 >> #####

```

Fonte: Elaborado pelo autor.

7.1 Resultados de Simulação

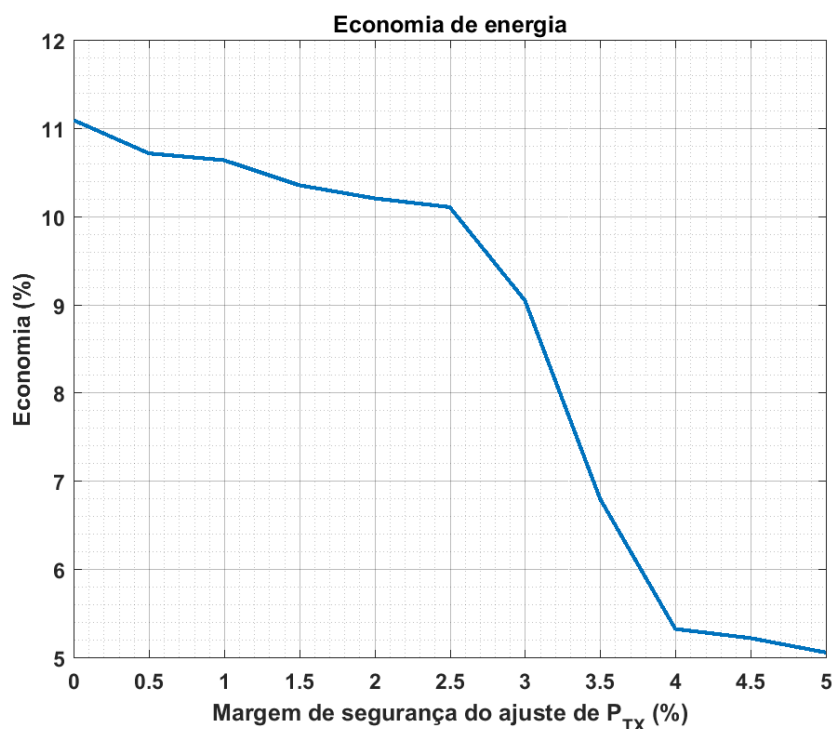
Os resultados de simulação exibidos neste tópico, seja para as métricas ou para o consumo total de energia, referem-se aos valores médios obtidos a partir de 5 rodadas de simulação. Em um primeiro momento foram obtidos os resultados para os protocolos DVR, AODV e DSR e, posteriormente, foi elencado o protocolo DSR para ser utilizado em conjunto com o algoritmo de ajuste de potência descrito no Capítulo 6. Este protocolo foi escolhido pois apresentou menor consumo total em relação aos outros protocolos, mantendo bom desempenho nas métricas calculadas, e será denominado DSR modificado daqui em diante.

7.1.1 Economia de Energia do Ajuste de Potência

O algoritmo de ajuste de potência de transmissão apresentado no Capítulo 6 foi adaptado dentro do código do protocolo de roteamento DSR e, por meio desse algoritmo, os nós que receberem dados de outros nós podem calcular um valor de potência de transmissão adequado para cada destino a partir do RSSI. Porém, para que o ajuste de potência seja feito com uma maior confiabilidade, decidiu-se somar ao valor calculado uma margem de segurança com valor entre 0 e 5%.

Sendo assim, foi avaliado o consumo total de energia em função da variação da margem de segurança considerada. Para este fim, foram feitas 3 rodadas de simulação para um cenário composto por 25 nós, sendo um nó sensor fonte, 23 nós roteadores e um *gateway*. Nesta simulação, o nó fonte manteve-se em movimento a uma velocidade fixa de 18 km/h e os demais nós da rede permaneceram estáticos; além disso, o nó sensor fonte enviou dados para o *gateway* a cada 30 segundos. Ao final, foi calculado o valor da economia de energia a partir do valor de consumo médio comparando as simulações do protocolo DSR com e sem ajuste de potência. A partir dos resultados para economia de energia mostrados na Figura 38, verificou-se que quando a margem de ajuste é maior que 2,5%, a economia de energia é inferior a 10%. Logo, considerando obter uma economia de energia de pelo menos 10%, foi decidido manter a margem de segurança no restante das simulações deste trabalho fixada em 2%.

Figura 38: Margem de segurança do ajuste de potência versus economia de energia.



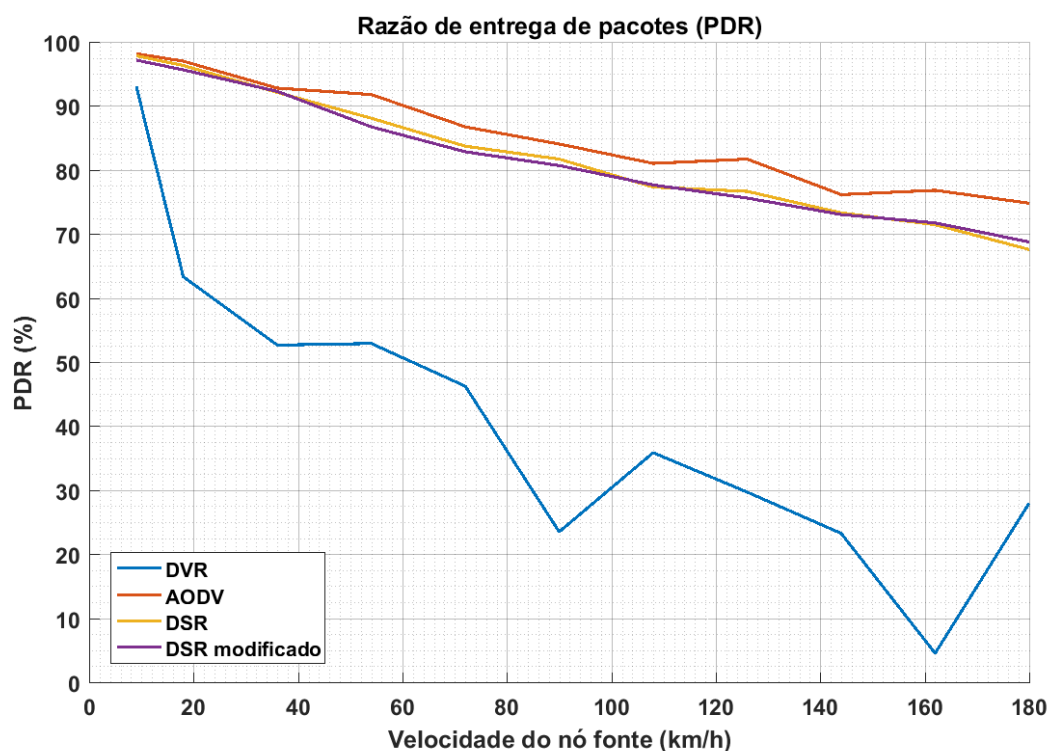
Fonte: Elaborado pelo autor.

7.1.2 Razão de Entrega de Pacotes

Os resultados de simulação em relação ao cálculo da métrica da razão de entrega de pacotes para todos os protocolos estão apresentados na Figura 39 e expostos com mais detalhes em tabelas no Apêndice A1. Os valores de desvio padrão e margem de erro para um nível de confiança de 95% também estão organizados nas tabelas.

Ao se observar os resultados apresentados na Figura 39, percebe-se que os protocolos reativos demonstraram melhor desempenho geral no envio de dados para o cenário de simulação considerado, sendo o protocolo AODV o protocolo com maior taxa de entrega de pacotes de dados, seguido do protocolo DSR para os casos com e sem ajuste de potência. Este resultado para o AODV pode ser explicado, pois esse protocolo possui um mecanismo de verificação de rotas que funciona de maneira periódica e também durante a movimentação do nó fonte, antecipando-se na descoberta de novas rotas para o caso de alguma indisponibilidade.

Figura 39: Resultados de simulação para a métrica PDR.



Fonte: Elaborado pelo autor.

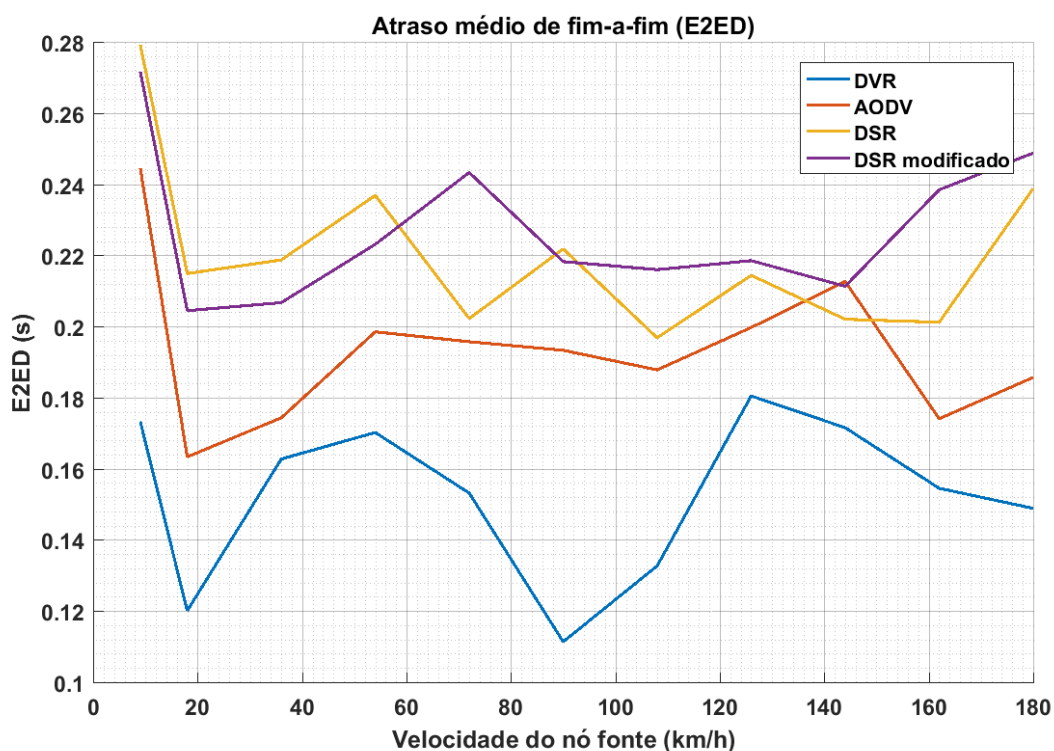
Para velocidades inferiores a 20 km/h, todos os protocolos mantiveram altas taxas de entrega de dados. Porém, após superar esse valor de velocidade, o decaimento de desempenho do protocolo DVR foi bastante elevado, levando-o, por exemplo, a perder mais de 50% dos pacotes transmitidos quando o nó fonte movimenta-se a velocidades superiores aos 60 km/h, enquanto que os demais protocolos mantinham taxas de entrega acima de 80% para a mesma faixa de velocidade. Dessa forma, o impacto da variação da velocidade do nó fonte foi muito maior no protocolo pró-ativo DVR, entretanto esta característica era prevista, pois novas rotas serão descobertas apenas dentro do intervalo fixo de roteamento de 5 minutos.

7.1.3 Atraso Médio Fim-a-Fim

Na Figura 40 estão ilustrados os resultados de desempenho dos protocolos de roteamento em relação ao atraso (ou latência) médio fim-a-fim, estando expostos no Apêndice A2 maiores detalhes. A partir destes resultados, observou-se que o protocolo DVR foi

o que demonstrou menor valor para latência, seguido pelo protocolo AODV. O protocolo DSR com ou sem ajuste de potência obteve maior latência média nas simulações em relação aos outros implementados.

Figura 40: Resultados de simulação para a métrica E2ED.



Fonte: Elaborado pelo autor.

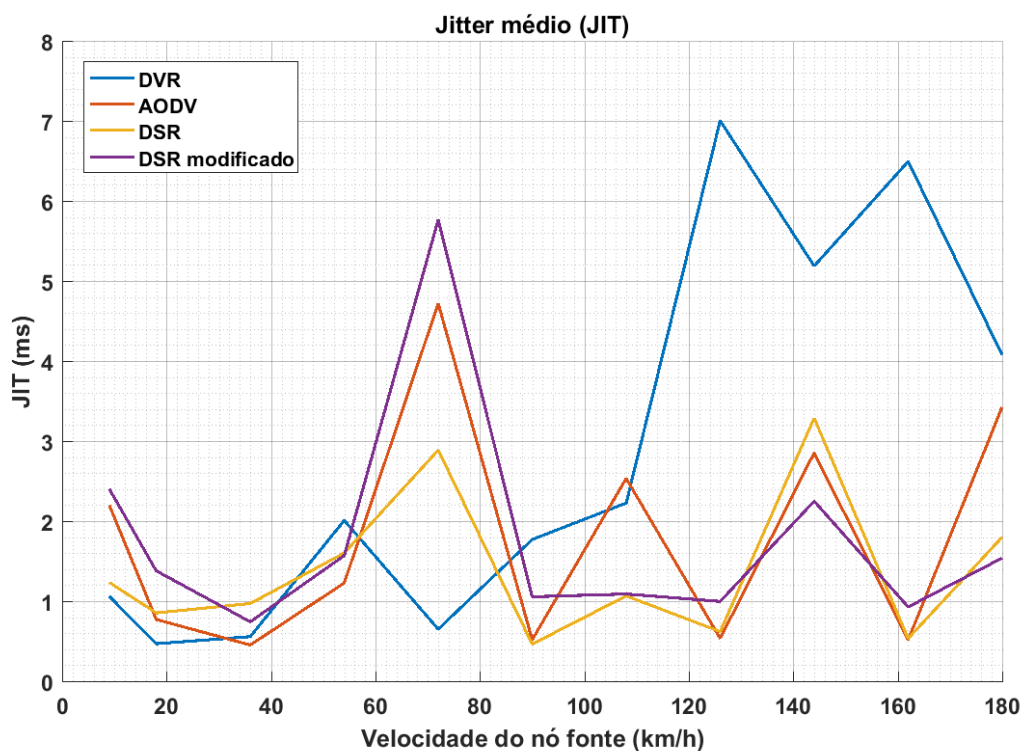
É importante destacar que o bom desempenho do protocolo DVR no quesito de latência é um comportamento esperado, por se tratar de um protocolo pró-ativo e independente da ocorrência de indisponibilidade de rotas. As descobertas dos novos caminhos para os destinos acontecem durante os períodos fixos de roteamento e ficam acessíveis nas tabelas antes da demanda de transmissão.

Alguns resultados semelhantes, obtidos por outros autores em experimentos anteriores, mostraram comportamento semelhante, embora os cenários de simulação fossem distintos. É o caso dos resultados apresentados por Rahman, Hasan e Islam (2012), em que foram calculadas as mesmas métricas e o protocolo DSR apresentou melhor desempenho para a métrica de atraso médio, enquanto que o AODV obteve um maior número de pacotes enviados com sucesso aos destinos.

7.1.4 Jitter Médio

O desempenho médio dos protocolos conforme a métrica do *jitter* pode ser observado nas tabelas do Apêndice A3 e está exibido na Figura 41. A partir destes dados, verificou-se que o protocolo DSR sem ajuste de potência de transmissão foi o único que não obteve uma variação para latência maior do que 4 ms. Para velocidades abaixo dos 100 km/h, o protocolo DVR apresentou comportamento semelhante, porém, no caso de altas velocidades torna-se o protocolo com maior *jitter*.

Figura 41: Resultados de simulação para a métrica JIT.



Fonte: Elaborado pelo autor.

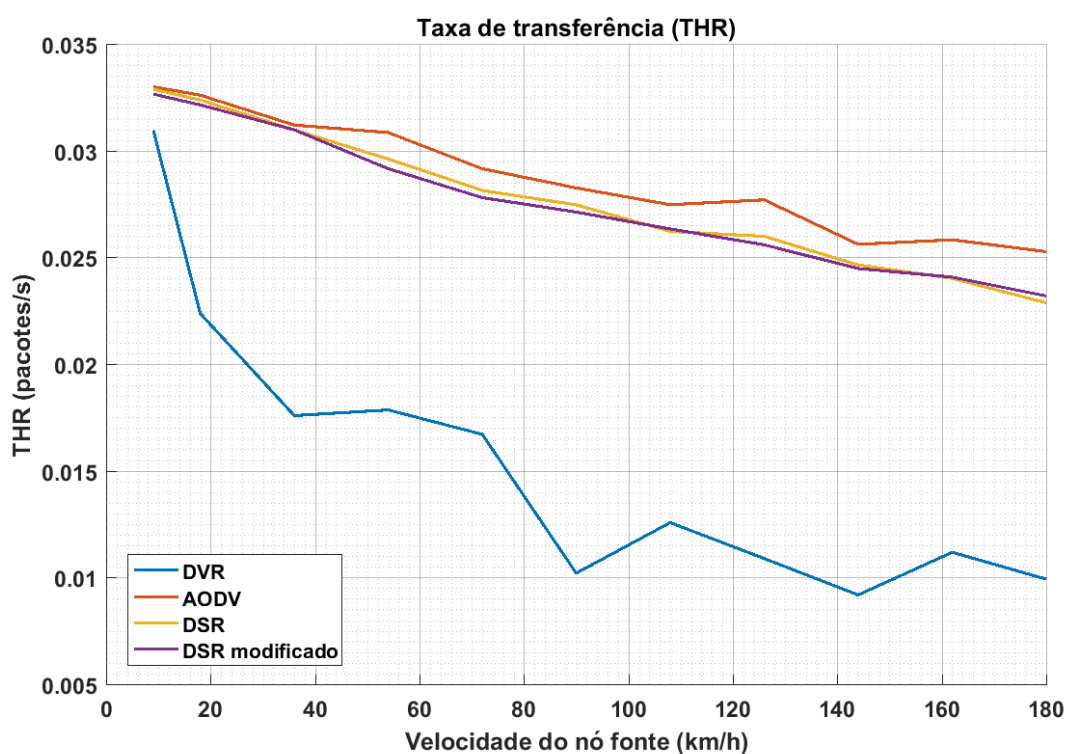
Estas situações de alta variação nos valores de latência podem ocasionar em uma não regularidade na entrega dos pacotes de dados, mesmo que as transmissões estejam sendo realizadas de maneira periódica, além de poder ser um indício de sobrecarga da rede.

7.1.5 Taxa de Transferência (*Throughput*)

No que diz respeito à taxa de transferência dos protocolos de roteamento, observou-se uma alta correlação com o comportamento da taxa de entrega de pacotes, estando os re-

sultados apresentados na Figura 42 e no Apêndice A4. Desta maneira, o bom desempenho do protocolo AODV em relação à razão de entrega de pacotes foi refletido na métrica de taxa de transferência, tornando este protocolo capaz de transmitir mais dados por unidade de tempo do que os demais. O protocolo DSR obteve um desempenho levemente inferior ao AODV nos casos em que foi considerado ou não o algoritmo de ajuste de potência de transmissão.

Figura 42: Resultados de simulação para a métrica THR.



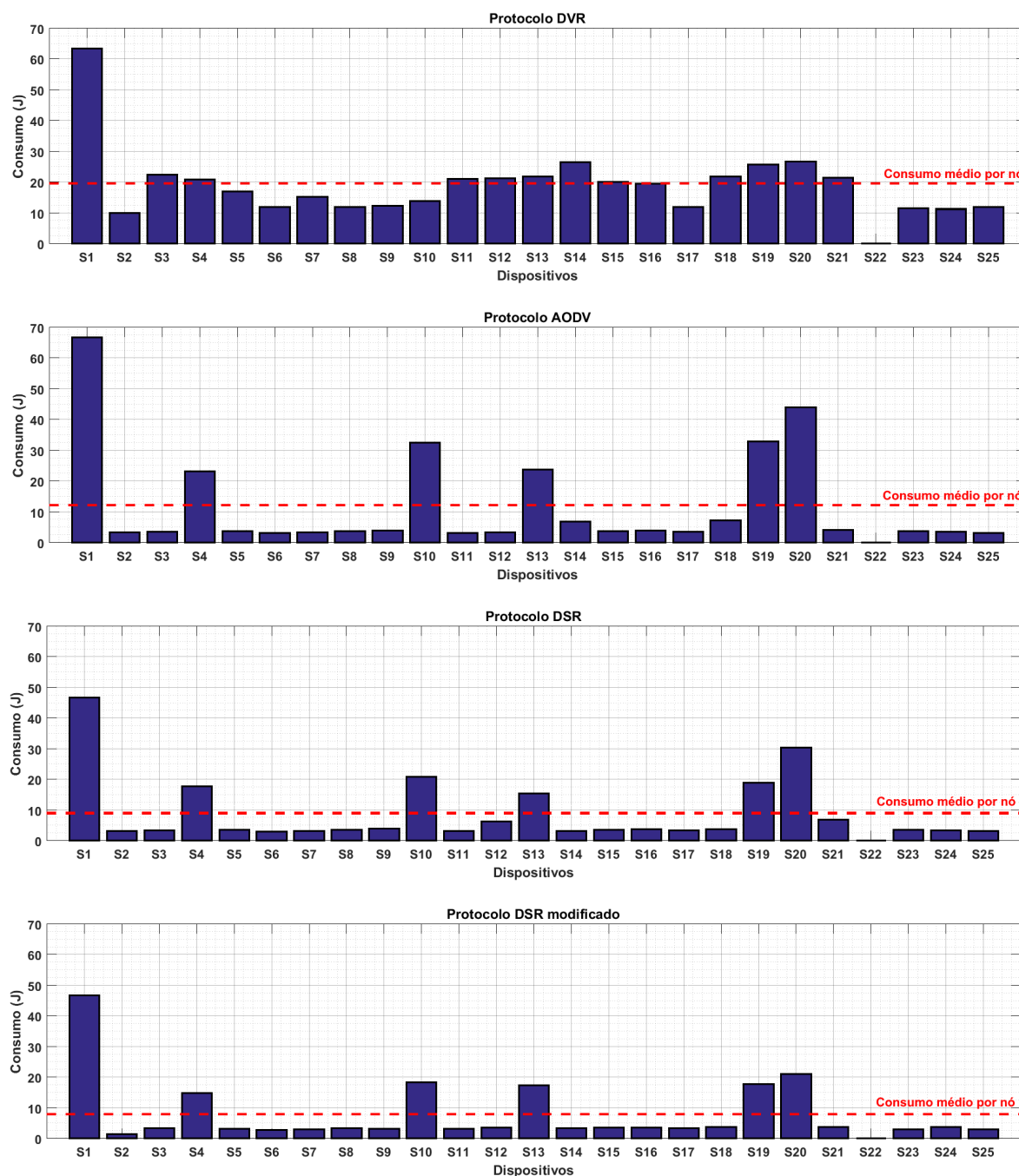
Fonte: Elaborado pelo autor.

Devido ao protocolo DVR obter informações relativas às rotas para os destinos apenas nos intervalos fixos de 5 minutos, este protocolo possui maior taxa de perda de pacotes quando submetido a altas velocidades, o que também ocasiona uma redução considerável na sua taxa de transferência, conforme pode ser observado na Figura 42.

7.1.6 Consumo Médio de Energia

Na Figura 43 estão exibidos os resultados de consumo médio de carga de bateria por nó em função do protocolo de roteamento.

Figura 43: Consumo médio de energia de cada nó por protocolo.



Fonte: Elaborado pelo autor.

Nas simulações para obtenção de consumo de carga de bateria, foi considerada para o nó sensor uma velocidade média fixa de 20 km/h, com base nos resultados obtidos para mobilidade de ônibus em centros urbanos realizados por Vianna *et al.* (2018), Marujo *et al.* (2015) e Rolnik e Klintowitz (2011).

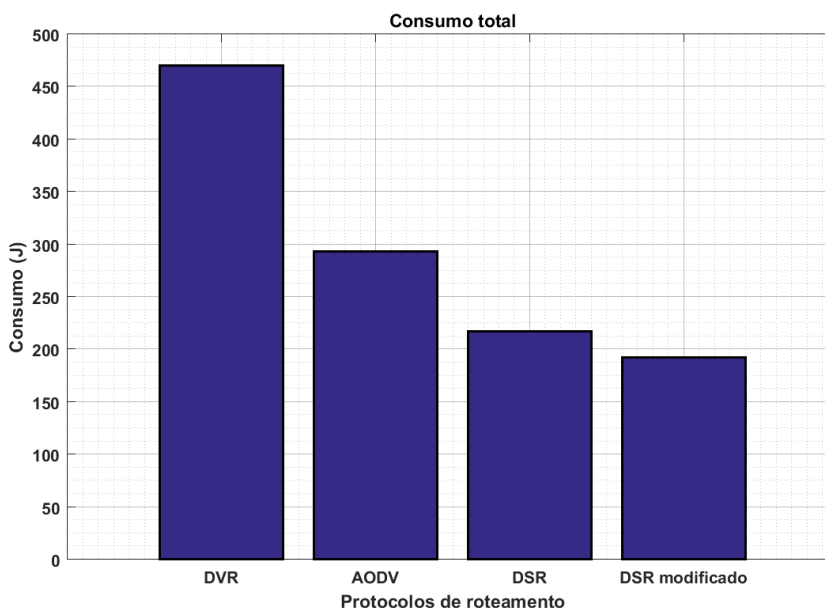
O consumo do nó S1 é maior em relação aos demais pois este é o nó fonte, responsável por realizar as medições e transmitir para o *gateway*, o qual está representado por S22. As curvas mostradas na Figura 43 permitem identificar quais nós da rede são mais utilizados em geral, sendo importante levar em consideração que o consumo de cada nó está relacionado ao protocolo utilizado e também à topologia de rede utilizada nas simulações.

Diante destes resultados, percebeu-se um comportamento mais distribuído no consumo de energia da rede para o protocolo DVR, enquanto que, no caso dos demais protocolos, o consumo manteve-se concentrado em alguns nós específicos. Isso ocorreu pois, durante o período de compartilhamento de tabelas de roteamento do protocolo DVR, é realizado um número muito maior de retransmissões de dados do que o que acontece no processo de descoberta de rotas dos protocolos reativos. Por outro lado, o consumo nas simulações dos protocolos reativos esteve concentrado nos nós S4, S10 e S13, estando estes nós localizados muito próximos das coordenadas geográficas das trajetórias do nó móvel, além dos nós S19 e S20, os quais possuem *link* de comunicação direta com o *gateway*.

Os resultados para consumo médio por nó em função do protocolo também são mostrados na Figura 43. O protocolo DVR apresentou um consumo médio por nó de aproximadamente 20 J, seguido do protocolo AODV com consumo de cerca de 12 J por nó. Os protocolos DSR com e sem ajuste de potência mostraram um consumo de 9 J e 8 J, respectivamente.

Para visualizar o consumo total por protocolo, os consumos individuais de cada nó foram somados, conforme exposto na Figura 44. A partir destes resultados, fica evidente o impacto do grande número de retransmissões no consumo total do protocolo DVR, levando-o a consumir cerca de 460 J ao final da simulação. Embora o protocolo AODV tenha sido o que apresentou melhor desempenho na entrega dos pacotes de dados, foi o segundo protocolo que mais consumiu energia, quase 300 J. A partir destes resultados, verificou-se que o protocolo DSR modificado foi o que consumiu menos energia ao longo dos 3600 segundos de simulação.

Figura 44: Consumo médio total por protocolo.



Fonte: Elaborado pelo autor.

Portanto, estes resultados comprovam a economia de energia proporcionada pelo algoritmo de ajuste de potência de transmissão proposto neste trabalho, levando o consumo médio total para um valor 11,32% menor em relação ao protocolo sem uso do algoritmo. Entretanto, este algoritmo ainda está em desenvolvimento e existem pendências a serem resolvidas em trabalhos futuros antes de ser possível implementá-lo experimentalmente.

7.2 Resultados Experimentais

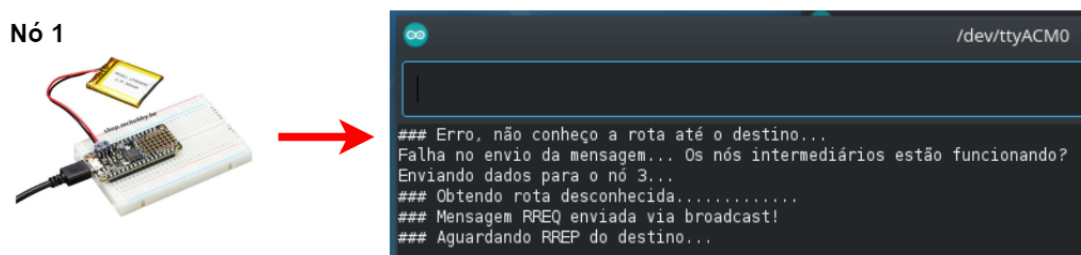
Neste tópico serão apresentados os resultados experimentais da rede que foi montada segundo a topologia proposta na Figura 34. O protocolo de roteamento DSR foi escolhido para ser utilizado na subrede em malha da topologia e foi implementado com adaptações baseado nos códigos de exemplo disponibilizados no site da biblioteca *RadioHead*.

7.2.1 Etapas de Roteamento

Inicialmente foi realizada a programação dos dispositivos na IDE para Arduino, adaptando-os para exibir as etapas de roteamento no *console*. Dessa forma, essa primeira etapa da montagem experimental consistiu em observar o funcionamento da rede com relação à descoberta de rotas. Após implementar o protocolo nos três nós da rede, apenas o nó fonte foi ligado e o processo de roteamento foi iniciado, conforme exibido na Figura 45.

Este primeiro nó foi programado para enviar a mensagem "Microengenharia" ao nó 3.

Figura 45: Teste experimental das etapas de roteamento com 1 nó.



Fonte: Elaborado pelo autor.

Em seguida, o segundo nó foi ligado e passou a receber as mensagens de requisição de rota (RREQ) enviadas pelo primeiro nó e encaminhá-las, porém, como o terceiro nó ainda não estava ligado, as mensagens RREQ não obtinham respostas (RREP). Como o protocolo DSR utiliza um roteamento de origem, ao receber uma requisição cada nó deverá adicionar o seu endereço na rota atual, sendo este valor exibido no *console* e identificado como rota acumulada. Este resultado está exposto a seguir:

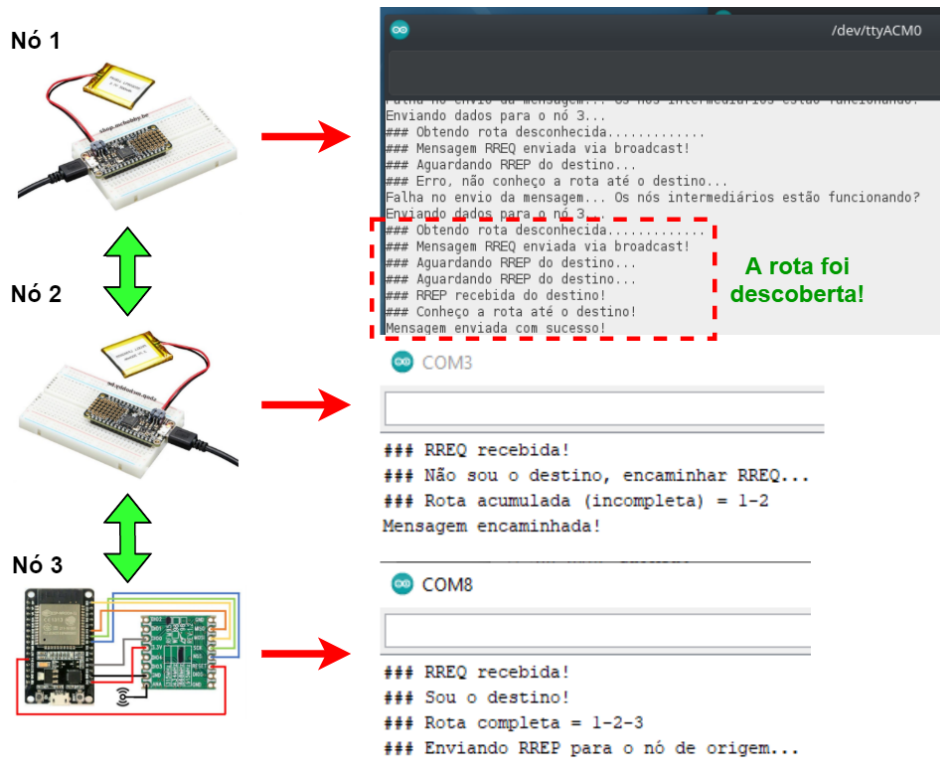
Figura 46: Teste experimental das etapas de roteamento com 2 nós.



Fonte: Elaborado pelo autor.

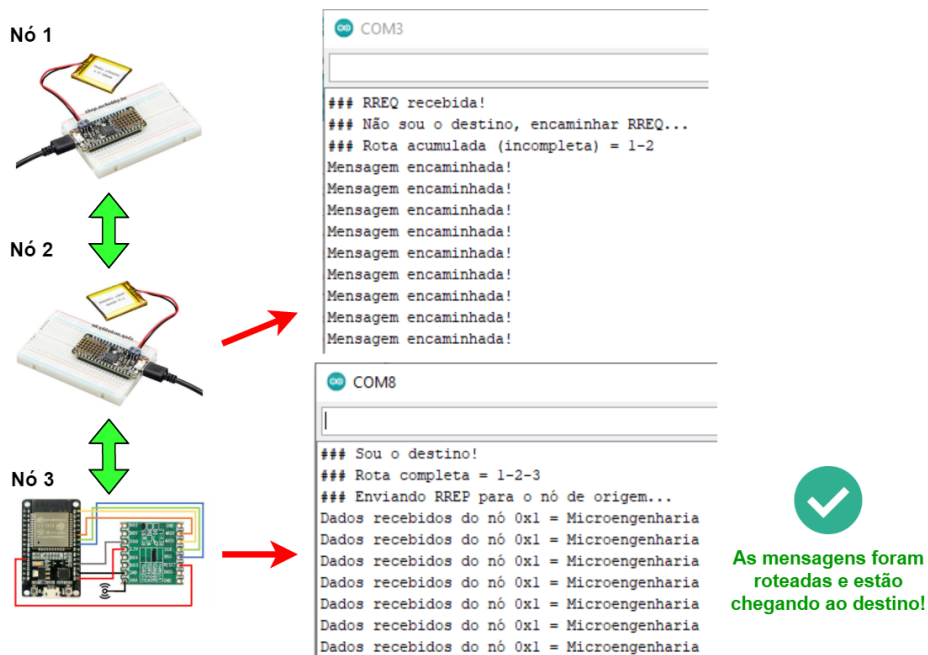
O último passo de verificação das etapas de roteamento deu-se ao ligar o terceiro nó, completando a rede e possibilitando fornecer respostas (RREP) para as requisições de rotas. Dessa maneira, após a execução da rotina de descoberta de rotas, as mensagens foram encaminhadas com sucesso a partir do nó 1 com destino ao nó 3. A comprovação do correto funcionamento das etapas de roteamento está ilustrado nas Figuras 47 e 48.

Figura 47: Teste experimental das etapas de roteamento com 3 nós.



Fonte: Elaborado pelo autor.

Figura 48: Confirmação das etapas de roteamento.

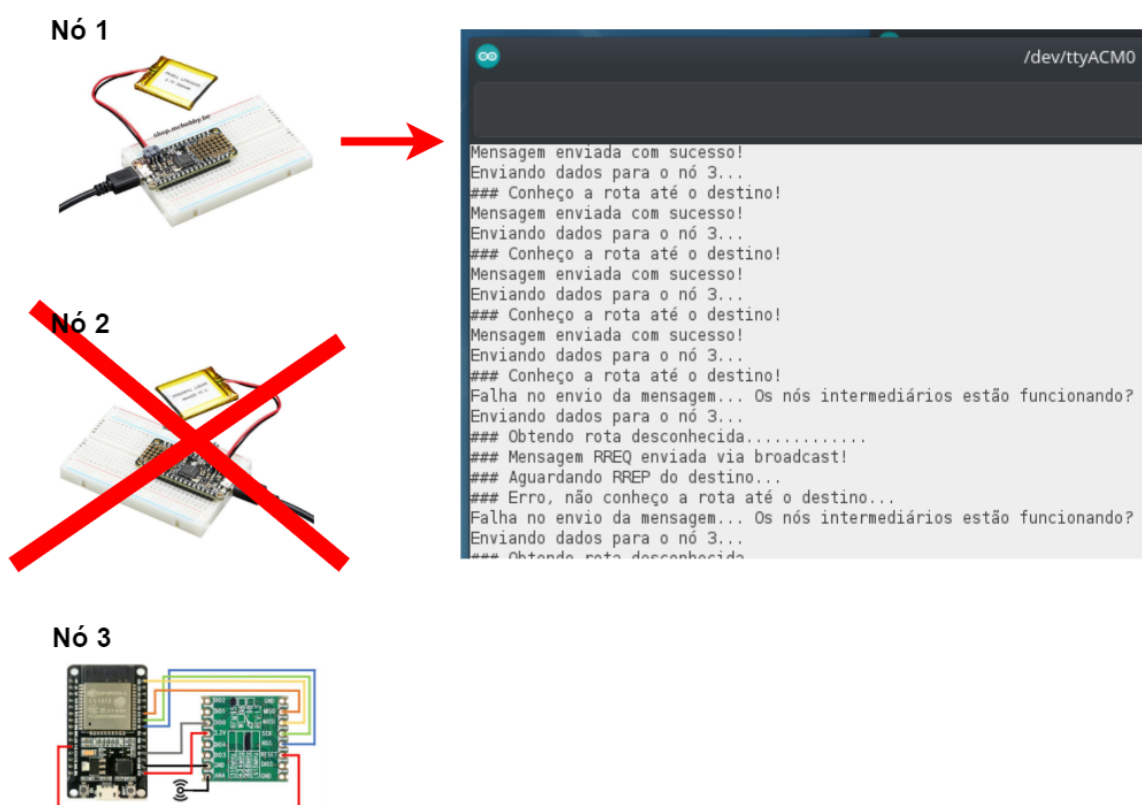


Fonte: Elaborado pelo autor.

Com o objetivo de verificar se o protocolo de roteamento implementado é capaz de obter novas rotas, o segundo nó foi desligado, tornando inválida, dessa forma, a rota existente. Assim que o primeiro nó detectou a indisponibilidade, o processo de requisição de rotas foi reiniciado, como exibido na Figura 49.

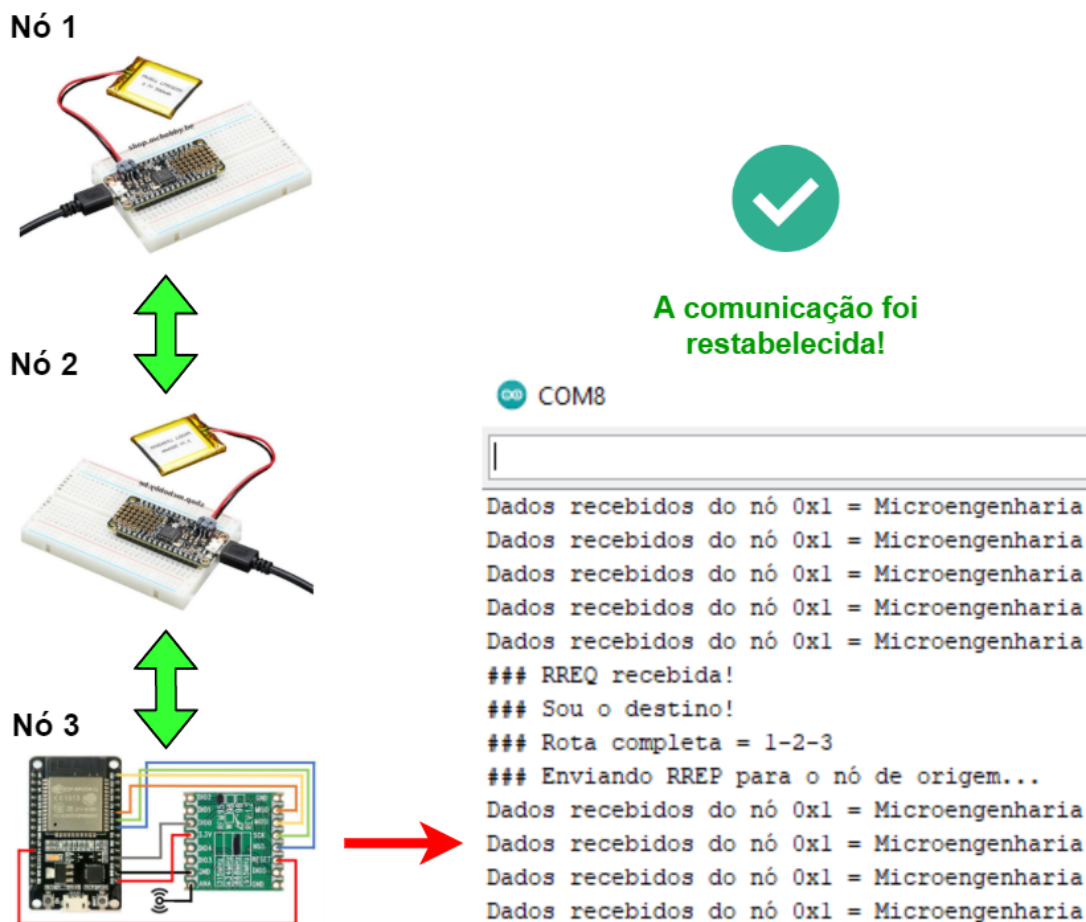
Por fim, o segundo nó foi ligado novamente e uma nova rota a partir do nó 1 até o nó 3 foi descoberta, restaurando o enlace de comunicação. Logo, as mensagens que foram transmitidas pelo nó 1 para o nó 3 podem ser observadas no *console* do nó 3 apresentado na Figura 50.

Figura 49: Teste de indisponibilidade de rota.



Fonte: Elaborado pelo autor.

Figura 50: Teste de indisponibilidade de rota.



Fonte: Elaborado pelo autor.

7.2.2 Teste Completo de Roteamento

Tendo sido comprovado o funcionamento de todas as etapas de roteamento do protocolo DSR, foi inserido no código do terceiro nó (concentrador) o algoritmo mostrado na Figura 36, habilitando o dispositivo para receber dados da rede em malha e encaminhá-los ao *gateway* utilizando o protocolo LoRaWAN. Além do mais, a estrutura da mensagem enviada pelo nó 1 (fonte) também foi alterada para o seguinte formado serializado:

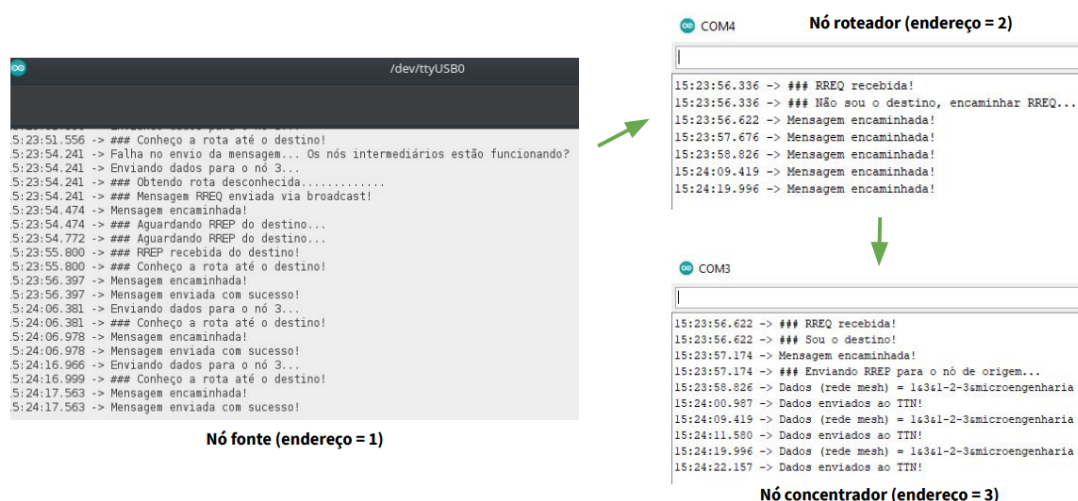
```
nó_fonte&nó_destino&rota_completa&microengenharia
```

Para que seja possível acompanhar a recepção dos dados no nó de destino, foi necessário criar uma aplicação no servidor *The Things Network* (TTN), servidor no qual está cadastrado o *gateway*. Nessa aplicação é necessário separar todas as variáveis dos dados que foram enviados pelo nó 1. Para isto, foi alterada a função de decodificação da aplicação do TTN no menu "*Payload Formats*", conforme exposto a seguir:

```
1 function Decoder(bytes , port){
2
3     // Inicializacao das variaveis
4     var origem = {};
5     var destino = {};
6     var rota_completa = {};
7     var dados = {};
8
9     // Converte os bytes recebidos em caracteres
10    msg_recebida_full = String.fromCharCode.apply(null , bytes);
11
12    // Separa as informacoes recebidas
13    msg_recebida = msg_recebida_full.split("&");
14    origem = msg_recebida[0];
15    destino = msg_recebida[1];
16    rota_completa = msg_recebida[2];
17    dados = msg_recebida[3];
18
19    dados = dados.substr(0, 15);
20
21    // Retorna para o console no formato JSON
22    return{
23        origem: origem ,
24        destino: destino ,
25        rota_completa: rota_completa ,
26        dados: dados
27    };
28
29 }
```

Após realizar todas as alterações, os três dispositivos da rede experimental foram ligados para envio dos dados ao *gateway* e os *consoles* de cada nó estão mostrados na Figura 51, comprovando o funcionamento do protocolo de roteamento e também o envio dos dados à aplicação do TTN.

Figura 51: *Console* dos nós para o teste completo de roteamento.



Fonte: Elaborado pelo autor.

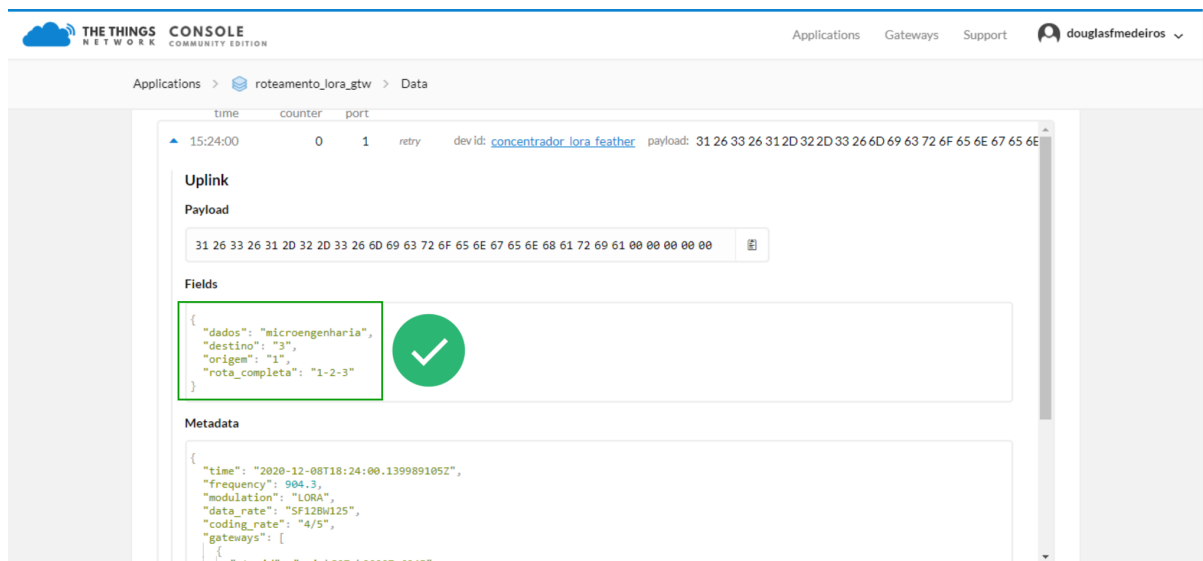
A confirmação de recebimento na aplicação do TTN das mensagens enviadas pelo nó 1 pode ser observada na Figura 52, enquanto que na Figura 53 estão exibidos os detalhes de uma dessas mensagens (com informações sobre o nó de destino, nó de origem, qual foi a rota pela qual os dados foram enviados e qual foi a mensagem de fato).

Figura 52: Confirmação de recebimento dos dados na aplicação do TTN.

Filters					
uplink	downlink	activation	ack	error	
time	counter	port			
15:33:17	0	1	retry	dev id: concentrador lora feather	payload: 31 26 33 26 31 2D 32 2D 33 26 6D 69 63 72 6F 65 6E 67 65 6E 64
15:33:06	0	1	retry	dev id: concentrador lora feather	payload: 31 26 33 26 31 2D 32 2D 33 26 6D 69 63 72 6F 65 6E 67 65 6E 64
15:32:56	0	1	retry	dev id: concentrador lora feather	payload: 31 26 33 26 31 2D 32 2D 33 26 6D 69 63 72 6F 65 6E 67 65 6E 64
15:32:45	0	1	retry	dev id: concentrador lora feather	payload: 31 26 33 26 31 2D 32 2D 33 26 6D 69 63 72 6F 65 6E 67 65 6E 64
15:32:35	0	1	retry	dev id: concentrador lora feather	payload: 31 26 33 26 31 2D 32 2D 33 26 6D 69 63 72 6F 65 6E 67 65 6E 64
15:32:24	0	1	retry	dev id: concentrador lora feather	payload: 31 26 33 26 31 2D 32 2D 33 26 6D 69 63 72 6F 65 6E 67 65 6E 64
15:32:13	0	1	retry	dev id: concentrador lora feather	payload: 31 26 33 26 31 2D 32 2D 33 26 6D 69 63 72 6F 65 6E 67 65 6E 64

Fonte: Elaborado pelo autor.

Figura 53: Detalhes de uma das mensagens recebidas no TTN.



The screenshot displays the 'Data' view for an application named 'roteamento_lora_gtw' in the The Things Network Console. The interface shows a table of data points with columns for time, counter, port, and a 'Data' column. The selected data point is at time '15:24:00', counter '0', and port '1', with a 'retry' status. The device ID is 'concentrador_lora_feather' and the payload is '31 26 33 26 31 2D 32 2D 33 26 6D 69 63 72 6F 65 6E 67 65 6E 68 61 72 69 61 00 00 00 00 00'. Below the table, the 'Uplink' section shows the 'Payload' as a hex string and the 'Fields' as a JSON object:

```
{  "dados": "microengenharia",  "destino": "3",  "origem": "1",  "rota_completa": "1-2-3"}
```

. A green checkmark icon is next to the fields. The 'Metadata' section shows a JSON object with details like time, frequency, modulation, data rate, coding rate, and gateways.

Applications > roteamento_lora_gtw > Data

time	counter	port	Data
15:24:00	0	1	retry dev id: concentrador_lora_feather payload: 31 26 33 26 31 2D 32 2D 33 26 6D 69 63 72 6F 65 6E 67 65 6E 68 61 72 69 61 00 00 00 00 00

Uplink

Payload

31 26 33 26 31 2D 32 2D 33 26 6D 69 63 72 6F 65 6E 67 65 6E 68 61 72 69 61 00 00 00 00 00

Fields

```
{  "dados": "microengenharia",  "destino": "3",  "origem": "1",  "rota_completa": "1-2-3"}
```

Metadata

```
{  "time": "2020-12-08T18:24:00.139989105Z",  "frequency": 904.3,  "modulation": "LORA",  "data_rate": "SF12BW125",  "coding_rate": "4/5",  "gateways": [    {      "gateway_id": "asif-h877ah00007ac745"    }  ]}
```

Fonte: Elaborado pelo autor.

8 Conclusões

Neste trabalho foram analisados protocolos de roteamento de duas categorias distintas: protocolos pró-ativos e reativos. Desta forma, foram implementados no simulador *Cupcarbon* os protocolos DVR, AODV e DSR. Além disso, também foi desenvolvida uma ferramenta auxiliar integrada à interface gráfica do *Cupcarbon* para calcular o raio de alcance entre os dispositivos de acordo com o modelo de propagação de Egli.

Conforme análise realizada, foi possível observar que os protocolos reativos apresentaram maiores taxas de entrega de pacotes, sendo o protocolo AODV o que obteve melhor desempenho em relação a esse critério, seguido do protocolo DSR. Estes resultados foram refletidos na taxa de transferência, mostrando uma alta correlação entre essas métricas. Assim, os protocolos reativos também obtiveram altas taxas de transferência para o cenário de simulação considerado.

Por outro lado, o protocolo DVR obteve um melhor desempenho no quesito de atraso médio, mas ao custo de perder muitos pacotes, mostrando-se ineficiente no caso do nó fonte mover-se em altas velocidades. Neste caso, de acordo com as simulações, este protocolo é mais adequado para situações em que os nós da rede permanecem estáticos. Além disso, com relação ao *jitter* médio, o protocolo DVR foi o que apresentou menores valores para velocidades de até 100 km/h, mas para velocidades superiores os protocolos reativos mostraram menor variação na latência.

Com os dados fornecidos pela ferramenta de simulação, também foi possível determinar o perfil de consumo energético de cada nó de acordo com o protocolo que está sendo utilizado. Logo, o protocolo que apresentou maior consumo médio total de energia foi o protocolo DVR, devido ao grande volume de retransmissões nas etapas de roteamento. O segundo protocolo que mais consumiu energia foi o AODV, seguido do protocolo DSR.

Diante dos resultados de simulação, o protocolo de roteamento DSR demonstrou ser o mais adequado para ser utilizado em conjunto com o algoritmo de ajuste de potência de transmissão proposto neste trabalho. Essa decisão foi tomada devido ao fato do DSR ter consumido menos energia, de uma maneira geral, quando comparado aos demais e mesmo assim ter mantido bons níveis para todas as métricas de avaliação consideradas. Além do mais, este protocolo é favorecido pela suas características de roteamento de origem, tornando-o livre do problema de *loops* infinitos de uma maneira relativamente

simples, assim como do seu funcionamento sob demanda, preservando a largura de banda de mensagens excessivas de controle. Assim sendo, o algoritmo de ajuste de potência de transmissão proposto nesta Dissertação proporcionou uma economia de energia de aproximadamente 11,32% com relação ao protocolo DSR quando simulado sem ajuste de potência, comprovando o seu benefício no tocante ao consumo de energia sem prejudicar a razão de entrega de pacotes.

Os resultados de simulação serviram como base para a implementação experimental que foi realizada neste trabalho. Para essa implementação foi definida uma topologia de rede mista, composta por uma subrede em malha que utilizou o protocolo de roteamento DSR em conjunto com a modulação LoRa e uma segunda parte com topologia em estrela de acordo com o protocolo LoRaWAN. Inicialmente foram validadas com sucesso todas as etapas de roteamento dos códigos programados nos dispositivos e, ao final, foi desenvolvida uma aplicação na nuvem para acompanhamento da recepção dos dados que foram roteados, sendo confirmado o correto funcionamento da rede.

Sendo assim, mesmo sendo uma ferramenta de simulação relativamente recente e em desenvolvimento, o simulador *Cupcarbon* mostrou-se capaz de proporcionar um ambiente de simulação adequado para aplicações em Cidades Inteligentes, apresentando como diferencial a possibilidade de utilização da tecnologia de comunicação sem fio LoRa nas simulações. Além disso, o *Cupcarbon* também fornece uma modelagem do consumo de energia, sendo muito importante observar esta questão dentro do contexto atual de aumento no número de dispositivos conectados em rede e trocando dados entre si.

8.1 Trabalhos Futuros

Diante dos resultados obtidos, foram elencadas algumas atividades futuras para continuação das atividades deste trabalho de Mestrado visando futuras pesquisas:

1. Implementação de outros protocolos de roteamento no simulador de redes *Cupcarbon*;
2. Desenvolvimento de outros modelos de propagação para uso nas definições de alcance entre os dispositivos, como por exemplo o modelo de log-distância;

3. Implementação e estudo de diferentes modelos de mobilidade aplicados ao contexto de Cidades Inteligentes, avaliando o desempenho dos protocolos;
4. Aprimoramento do algoritmo de ajuste de potência, visando aumentar o nível de economia geral proporcionado e possibilitar a sua implementação experimental;
5. Avaliação de protocolos de roteamentos híbridos;
6. Otimização no código dos protocolos de roteamento visando equalizar o consumo de energia dos nós da rede; e
7. Realização de testes experimentais com maior número de nós e por um maior período de tempo.

8.2 Publicações

Ao longo deste trabalho, os resultados obtidos durante esta Dissertação de Mestrado resultaram nas seguintes publicações:

1. RODRIGUES, Vitor José Costa et al. LoRa System for Monitoring and Facial Recognition. **Journal of Communication and Information Systems (JCIS)**, v. 36, n. 1, p. 1–16, 2021. ISSN 19806604. Disponível em: <<https://jcis.sbrt.org.br/jcis/article/view/745>>.
2. MEDEIROS, Douglas de Farias et al. Implementation and Analysis of Routing Protocols for LoRa Wireless Mesh Networks. **11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)**. IEEE, 2020.
3. MEDEIROS, Douglas de Farias et al. Análise Comparativa entre as Tecnologias LoRa, Zigbee e Wi-Fi utilizando o simulador Cupcarbon. **X Conferência Nacional em Comunicações, Redes e Segurança da Informação (ENCOM)**, 2020.
4. VILLARIM, Mariana Rodrigues et al. Calibration of a Low Cost Sensor for PM2.5 using a Reference PM Monitoring Station. **6th International Conference on Sensors and Electronic Instrumentation Advances (SEIA)**. IEEE, 2020.

5. MEDEIROS, Douglas de Farias *et al.* Avaliação da Modulação LoRa Utilizando Rádio Definido por Software. **IX Conferência Nacional em Comunicações, Redes e Segurança da Informação – ENCOM**, p. 143–144, 2019.
6. VILLARIM, Mariana Rodrigues *et al.* An Evaluation of LoRa Communication Range in Urban and Forest Areas: A Case Study in Brazil and Portugal. **2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference, IEMCON 2019**, p. 827–832, 2019. IEEE.
7. VILLARIM, Mariana Rodrigues *et al.* LoRa performance assessment in dense urban and forest areas for environmental monitoring. **INSCIT 2019 - 4th International Symposium on Instrumentation Systems, Circuits and Transducers**, 2019. IEEE.
8. DE LUNA, João Vitor Holanda *et al.* Compu-tation-time and mean-squared-error trade-off of calibration curves for electrical soil conductivity smart sensor. **INSCIT 2019 - 4th International Symposium on Instrumentation Systems, Circuits and Transducers**, 2019. IEEE.
9. LIMA, Diana Bezerra Correia *et al.* A Performance Evaluation of Raspberry Pi Zero W Based Gateway Running MQTT Broker for IoT. **2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference, IEMCON 2019**, p. 76–81, 2019. IEEE.

Referências

- ADAFRUIT. Datasheet Adafruit Feather M0 Radio with LoRa Radio Module. 2019. Disponível em: <<https://cdn-learn.adafruit.com/downloads/pdf/adafruit-feather-m0-radio-with-lora-radio-module.pdf>>.
- ADAFRUIT. Adafruit TinyLoRa Library. 2020. Disponível em: <<https://github.com/adafruit/TinyLoRa>>.
- AGUIAR, L. *Uma análise de modelamento de canal e confiabilidade em redes de sensores sem fio em linha*. João Pessoa: Universidade Federal da Paraíba, 2018.
- AHMAD, N. *et al.* IOT based wireless sensor network for precision agriculture. *IEEECON 2019 - 7th International Electrical Engineering Congress, Proceedings*, IEEE, p. 19–22, 2019.
- AKYILDIZ, I. F. *et al.* A Survey on Sensor Networks. *IEEE Communications Magazine*, 2002. ISSN 1536-1284.
- AL-DHIEF, F. T. *et al.* MANET Routing Protocols Evaluation: AODV, DSR and DSDV Perspective. *MATEC Web of Conferences*, v. 150, n. January, 2018. ISSN 2261236X.
- ALMEIDA, N. C. *et al.* Proposal of a Hybrid LoRa Mesh / LoRaWAN Network. *2020 IEEE International Workshop on Metrology for Industry 4.0 and IoT, MetroInd 4.0 and IoT 2020 - Proceedings*, p. 702–707, 2020.
- ASTHON, K. That ' Internet of Things ' Thing. *RFID Journal*, p. 4986, 2010. ISSN 00280836. Disponível em: <<http://www.rfidjournal.com/article/print/4986>>.
- AUGUSTIN, A. *et al.* A study of Lora: Long range & low power networks for the internet of things. *Sensors (Switzerland)*, v. 16, n. 9, p. 1–18, 2016. ISSN 14248220.
- BARIZON, B.-H. M. *Medidas de Propagação em 2.4 Ghz para o Planejamento de Redes Locais de Acesso sem fio*. Rio de Janeiro: Pontifícia Universidade Católica do Rio de Janeiro, 2004.
- BAYRAKDAR, M. E. Exploiting cognitive wireless nodes for priority-based data communication in terrestrial sensor networks. *ETRI Journal*, v. 42, n. 1, p. 36–45, 2 2020. ISSN 1225-6463. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.4218/etrij.2019-0296>>.
- BECKER, J. L. *Estatística basica: transformando dados em informacao*. Porto Alegre - RS: Bookman, 2015. ISBN 978-85-8260-313-0.
- BHANU, K. N.; JASMINE, H. J.; MAHADEVASWAMY, H. S. Machine learning Implementation in IoT based Intelligent System for Agriculture. In: *2020 International Conference for Emerging Technology (INCET)*. Belgaum: IEEE, 2020. p. 1–5. ISBN 9781728162218.
- BHATT, J.; HEMRAJANI, N. Effective Routing Protocol (DSDV) for Mobile Ad Hoc Network. *International Journal of Soft Computing and Engineering (IJSCE)*, v. 3, n. 5, 2013. ISSN 22312307.

BOUKERCHE, A. *Algorithms and Protocols for Wireless and Mobile Ad Hoc Networks*. [S.l.: s.n.], 2009.

BOUNCEUR, A. *et al.* CupCarbon: A new platform for the design, simulation and 2D/3D visualization of radio propagation and interferences in IoT networks. *CCNC 2018 - 2018 15th IEEE Annual Consumer Communications and Networking Conference*, v. 2018-Janua, p. 1–4, 2018.

BOUNCEUR, A.; MEHDI, K.; LOUNIS, M. A CupCarbon Tool for Simulating Destructive Insect Movements. *1st IEEE International Conference on Information and Communication Technologies for Disaster Management (ICT-DM'14)*, n. March, p. 24–25, 2014. Disponível em: <<http://hal.univ-brest.fr/hal-00963927{\%}5Cnhttps://hal-institut-mines-telecom.archives-ouvertes.fr/hal-009639>>.

BUCHHEIM, T. *NAM: Network Animator*. 2002. Disponível em: <<https://www.isi.edu/nsnam/nam/index.html>>.

CASTELLANOS, P. V. G. *Modelamento de Efeitos de Multipercurso Atmosférico em Enlaces Microondas em Visibilidade*. Rio de Janeiro: Pontifícia Universidade Católica do Rio de Janeiro, 2003.

CENTENARO, M. *et al.* Long-range communications in unlicensed bands: The rising stars in the IoT and smart city scenarios. *IEEE Wireless Communications*, Institute of Electrical and Electronics Engineers Inc., v. 23, n. 5, p. 60–67, 10 2016. ISSN 15361284.

CORKE, P. *et al.* Environmental wireless sensor networks. *Proceedings of the IEEE*, v. 98, n. 11, p. 1903–1917, 2010. ISSN 00189219.

DEBBARMA, M. K.; SEN, S. K.; ROY, S. A Review of DVR-based Routing Protocols for Mobile Ad Hoc Networks. *International Journal of Computer Applications*, v. 58, n. 3, p. 1–8, 2012.

DIJKSTRA, E. W. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik 1*, p. 269–271, 1959.

Elprocus. *ZigBee Wireless Technology Architecture and Applications*. 2020. Disponível em: <<https://www.elprocus.com/what-is-zigbee-technology-architecture-and-its-applications/>>.

ELSHABRAWY, T.; ROBERT, J. Closed-Form Approximation of LoRa Modulation BER Performance. *IEEE Communications Letters*, IEEE, v. 22, n. 9, p. 1778–1781, 2018. ISSN 15582558.

EMA, R. R. *et al.* Simulation Based Performance Analysis of Proactive, Reactive and Hybrid Routing Protocols in Wireless Sensor Network. *2020 11th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2020*, 2020.

FARIAS, M. M. *Protocolo de roteamento para redes wireless mesh*. Porto Alegre: Pontifícia Universidade Católica do Rio Grande do Sul, 2008.

GAWIY, M. *et al.* Performance Analysis of Destination Sequenced Distance Vector Routing Protocol in MANET. *International Journal of Ad Hoc, Vehicular and Sensor Networks*, 2014.

GOMES, J. A. P. *et al.* Smart Cities: Construção Sustentável E Edifícios Inteligentes São Tendências Para O Futuro. *Brazilian Journal of Development*, v. 6, n. 10, p. 76465–76484, 2020. ISSN 25258761.

HARINDA, E. *et al.* Comparative Performance Analysis of Empirical Propagation Models for LoRaWAN 868MHz in an Urban Scenario. *IEEE 5th World Forum on Internet of Things, WF-IoT 2019 - Conference Proceedings*, IEEE, p. 154–159, 2019.

HEDRICK, C. *Routing Information Protocol*. USA, 1988. Disponível em: <<https://www.rfc-editor.org/info/rfc1058>>.

HEIDEMANN, J.; MILLS, K.; KUMAR, S. Expanding confidence in network simulations. *IEEE Network*, v. 15, n. 5, p. 58–63, 2001. ISSN 08908044.

HEINZELMAN, W. B.; CHANDRAKASAN, A. P.; BALAKRISHNAN, H. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, IEEE, v. 1, n. 4, p. 660–670, 2002. ISSN 15361276.

HEINZELMAN, W. R.; CHANDRAKASAN, A.; BALAKRISHNAN, H. Energy-Efficient Communication Protocols for Wireless Microsensor Networks. *Proceedings of the 33rd Hawaiian International Conference on Systems Science (HICSS)*, 2000.

JOHNSON, D. B.; MALTZ, D. A. Dynamic Source Routing in Ad Hoc Wireless Networks. In: IMIELINSKI, T.; KORTH, H. (Ed.). *Mobile Computing*. [S.l.]: Kluwer Academic Publishers, 1996. cap. 5, p. 153–181.

KUMAR, A. *Introduction to LoRa Technology*. 2019. Disponível em: <<http://www.embien.com/blog/introduction-to-lora-technology/>>.

LEE, H. C.; KE, K. H. Monitoring of Large-Area IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation. *IEEE Transactions on Instrumentation and Measurement*, IEEE, v. 67, n. 9, p. 2177–2187, 2018. ISSN 00189456.

LoRa Alliance. *What is LoRaWAN® Specification*. 2021. Disponível em: <<https://loro-alliance.org/about-lorawan/>>.

LUETH, K. L. *State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating*. 2018. Disponível em: <<https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/>>.

LUNDELL, D. *et al.* A Routing Protocol for LoRA Mesh Networks. *19th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2018*, IEEE, 2018.

MACARAEG, K. C. V. G.; HILARIO, C. A. G.; AMBATALLI, C. D. C. LoRa-based Mesh Network for Off-grid Emergency Communications. *2020 IEEE Global Humanitarian Technology Conference (GHTC) 2020*, p. 1–4, 2020.

MALOKU, H. *et al.* Path loss model fitting for TV bands based on experimental measurements for urban environments in Kosovo. *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2019 - Proceedings*, p. 480–485, 2019.

MANNAN, P.; JAYAVIGNESH, T. Alternate simplistic approach to solve count-To-infinity problem by introducing a new flag in the routing table. *Proceedings - 2016 2nd IEEE International Conference on Research in Computational Intelligence and Communication Networks, ICRCICN 2016*, IEEE, p. 291–295, 2017.

MARDENI, R.; PEY, L. Y. Path loss model development for urban outdoor coverage of Code Division Multiple Access (CDMA) system in Malaysia. In: *2010 International Conference on Microwave and Millimeter Wave Technology*. IEEE, 2010. v. 6, n. 1, p. 441–444. ISBN 978-1-4244-5705-2. ISSN 19131844. Disponível em: <<http://ieeexplore.ieee.org/document/5525001/>>.

MARTINS, G. N. *et al.* Análise Comparativa entre Testbed e Simulador em Redes de Sensores Sem Fio IEEE 802.15.4 com Unslotted CSMA/CA. *Revista de Tecnologia da Informação e Comunicação*, v. 7, n. 2, p. 41–48, 2017.

MARUJO, L. G. *et al.* Um Método Para Avaliação Do Desempenho Do Serviço De Ônibus Baseado Em Dados De Gps. *XXIX Congresso Nacional de Pesquisa em Transporte da ANPET*, n. 2011, p. 1194–1205, 2015.

MATRE, V.; KARANDIKAR, R. Multipath routing protocol for mobile adhoc networks. *2016 Symposium on Colossal Data Analysis and Networking, CDAN 2016*, IEEE, p. 0–4, 2016.

MCCAULEY, M. RadioHead Library Examples. 2020. Disponível em: <<http://www.airspayce.com/mikem/arduino/RadioHead/examples.html>>.

MCCAULEY, M. RadioHead Packet Radio library for embedded microprocessors. 2020. Disponível em: <<http://www.airspayce.com/mikem/arduino/RadioHead/index.html>>.

MEDEIROS, D. d. F. *et al.* Implementation and Analysis of Routing Protocols for LoRa Wireless Mesh Networks. In: *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, 2020. p. 0020–0025. ISBN 978-1-7281-8416-6. Disponível em: <<https://ieeexplore.ieee.org/document/9284888/>>.

MEDHI, D.; RAMASAMY, K. *Network Routing : Algorithms, Protocols and Architectures*. [S.l.]: Elsevier, 2007.

MEHDI, K. *et al.* CupCarbon: A multi-agent and discrete event Wireless Sensor Network design and simulation tool. *SIMUTools 2014 - 7th International Conference on Simulation Tools and Techniques*, n. March, p. 126–131, 2014.

MOCHIZUKI, K. *et al.* Development and field experiment of wide area Wi-SUN system based on IEEE 802.15.4g. *2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016*, IEEE, p. 76–81, 2017.

NETO, J. A. T. C. Análise de Protocolos de Roteamento Unicast em Redes Ad Hoc Móveis Baseada em um Modelo Realístico de Bateria. 2011.

NSNAM. *The Network Simulator - NS-2*. 2020. Disponível em: <<https://www.isi.edu/nsnam/ns/>>.

ODUNLADE, E. *Introduction to LoRa and LoRaWAN: What is LoRa and How Does It Work?* 2019. Disponível em: <<https://circuitdigest.com/article/introduction-to-lora-and-lorawan-what-is-lora-and-how-does-it-work>>.

OKUMURA, Y. *et al.* Field strength and its variability in UHF and VHF land-mobile radio service. *Rev. Elec. Commun. Lab.*, v. 16, 1968.

ONIPE, J. A. *et al.* Optimal Propagation Models for Path-loss Prediction in a Mountainous Environment at 2100MHz. *2020 International Conference in Mathematics, Computer Engineering and Computer Science, ICMCECS 2020*, v. 231, 2020.

OPENSIM. *What is OMNeT++?* 2019. Disponível em: <<https://omnetpp.org/intro/>>.

PADIYA, S. D. *A System for MANET to detect Selfish Nodes using NS2*. Tese (Doutorado) — Patel College of Science & Technology, 2013.

PARVATHI, P. *Comparative analysis of CBRP, AODV, DSDV routing protocols in mobile Ad-hoc networks*. 2012.

PENG, J. Radio Propagation Models in Wireless Networks of Unmanned Aerial Vehicles. *International journal of Computer Networks & Communications*, v. 7, n. 3, p. 119–126, 2015. ISSN 09752293.

PERKINS, C. E.; BHAGWAT, P. Highly Dynamic Destination-Sequenced Distance-Vector Routing for Mobile Computers. *Sigcomm*, p. 234–244, 1994.

PERKINS, C. E.; ROYER, E. M. Ad-hoc on-demand distance vector routing. *Proceedings - WMCSA'99: 2nd IEEE Workshop on Mobile Computing Systems and Applications*, IEEE, p. 90–100, 1999.

PHAM, C. *et al.* Radio channel access challenges in LoRa low-power wide-area networks. INC, 2020. 65–102 p. ISBN 9780128188804. Disponível em: <<http://dx.doi.org/10.1016/B978-0-12-818880-4.00004-1>>.

QARAQE, M. *et al.* Performance analysis of chirp spread spectrum system under mobility scenario. *Physical Communication*, Elsevier B.V., v. 43, p. 101233, 2020. ISSN 18744907. Disponível em: <<https://doi.org/10.1016/j.phycom.2020.101233>>.

RAHMAN, J.; HASAN, M. A. M.; ISLAM, M. K. B. Comparative analysis the performance of AODV, DSDV and DSR routing protocols in wireless sensor network. *2012 7th International Conference on Electrical and Computer Engineering, ICECE 2012*, IEEE, p. 283–286, 2012.

RAPPAPORT, T. *Wireless Communications: Principles and Practice*. 2nd. ed. USA: Prentice Hall PTR, 2001. ISBN 0130422320.

RAZA, U.; KULKARNI, P.; SOORIYABANDARA, M. Low Power Wide Area Networks: An Overview. *IEEE Communications Surveys and Tutorials*, Institute of Electrical and Electronics Engineers Inc., v. 19, n. 2, p. 855–873, 4 2017. ISSN 1553877X.

REBELATO, J. G. B. *Internet das coisas para medição inteligente de energia com ênfase em tecnologias de comunicação*. Curitiba - PR: Universidade Tecnológica Federal do Paraná – UTFPR, 2018.

RIVERBED. *Riverbed Modeler: A Suite of Protocols and Technologies with a Sophisticated Development Environment*. 2020. Disponível em: <<https://www.riverbed.com/gb/products/steelcentral/steelcentral-riverbed-modeler.html>>.

ROCHOL, J. et al. *Plataformas de Simulação de Software Livre para Redes Fixas e Móveis: Características, Suporte, Instalação e Validação*. 2003.

RODRIGUES, V. et al. LoRa System for Monitoring and Facial Recognition. *Journal of Communication and Information Systems*, v. 36, n. 1, p. 1–16, 2021. ISSN 19806604. Disponível em: <<https://jcis.sbrt.org.br/jcis/article/view/745>>.

ROLNIK, R.; KLINTOWITZ, D. Mobilidade na cidade de São Paulo. *Estudos Avançados*, v. 25, n. 71, p. 89–108, apr 2011. ISSN 0103-4014. Disponível em: <http://www.scielo.br/scielo.php?script=sci{_}arttext{&}pid=S0103-40142011000100007{&}lng=p>.

ROSHAN, R.; MISHRA, S.; MEHER, C. P. A comparison between the AODV and DSDV routing protocols for mobile Ad-Hoc Network Using NS2. *Proceedings of the 2019 6th International Conference on Computing for Sustainable Global Development, INDIACom 2019*, p. 286–289, 2019.

ROZORINOV, H. et al. Features virtualization software telecommunications networks by means of simulation Riverbed Modeler. In: *2017 2nd International Conference on Advanced Information and Communication Technologies (AICT)*. IEEE, 2017. p. 246–249. ISBN 978-1-5386-0637-7. Disponível em: <<http://ieeexplore.ieee.org/document/8020111/>>.

RUIZ, L. B. et al. *Arquiteturas para Redes de Sensores Sem Fio*. [S.l.]: SBC Brazilian Computer Society, 2004.

SAMPOORNAM, K. P.; RAAGA, D. G. Performance Analysis of Bellman Ford, AODV, DSR, ZRP and DYMO Routing Protocol in MANET using EXATA. *Proceedings of the 2019 International Conference on Advances in Computing and Communication Engineering, ICACCE 2019*, 2019.

SANTOS, A. C. P. et al. Integração de Redes de Sensores sem Fio à Plataforma Lora ® para Serviços de Monitoramento e Controle Inteligentes. In: *XXXVIII SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES E PROCESSAMENTO DE SINAIS - SBrT 2020*. Florianópolis - SC: [s.n.], 2020.

SARAIVA, F. A. M.; BISPO, A.; GOMES, R. D. Avaliação da Confiabilidade de uma Rede Wi-SUN com Topologia Linear para Aplicações de Cidades Inteligentes. *XXXVIII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais - SBrT*, p. 22–25, 2020.

SCULLY, P. *Top 10 IoT applications in 2020*. 2020. Disponível em: <<https://iot-analytics.com/top-10-iot-applications-in-2020/>>.

SEMTECH. Datasheet Sx1276/77/78/79. Rev. 4. n. March, p. 131, 2015. Disponível em: <www.semtech.com>.

Semtech. *What is LoRa?* 2020. Disponível em: <<https://www.semtech.com/technology/lora/what-is-lora>>.

SHARMA, P. K.; SHARMA, D.; SAI, T. V. Optimization of propagation path loss model in 4G wireless communication systems. *Proceedings of the 2nd International Conference on Inventive Systems and Control, ICISC 2018*, IEEE, v. 20, n. Icisc, p. 1245–1248, 2018.

SILVA, J. C. da. *Influência da Vegetação no Desvanecimento e na Perda de Percurso de Enlaces de Radiocomunicação UHF na Faixa de 700 MHz*. Rio de Janeiro: Pontifícia Universidade Católica do Rio de Janeiro, 2014. Disponível em: <https://www.maxwell.vrac.puc-rio.br/24783/24783_4.PDF>.

SKLAR, B. *Digital Communications: Fundamentals & Applications, 2nd ed.* [S.l.]: Pearson Education, 2009. ISBN 9788131720929.

SRIVASTAVA, A. *et al.* Survey and overview of Mobile Ad-Hoc Network routing protocols. *2014 International Conference on Advances in Engineering and Technology Research, ICAETR 2014*, IEEE, p. 5–10, 2014.

Statista Analytics. *IoT: number of connected devices worldwide 2012-2025*. 2016. Disponível em: <<https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/#:~:text=InternetofThings-numberofconnecteddevices,by2025%2Cafivefoldincreaseintenyyears.>>>

TADAYONI, R.; HENTEN, A.; FALCH, M. Internet of Things - The battle of standards. In: *Joint 13th CTTE and 10th CMI Conference on Internet of Things - Business Models, Users, and Networks*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2017. v. 2018-January, p. 1–7. ISBN 9781538631973.

TANENBAUM, A. S. *Redes de Computadores*. 5. ed. [S.l.]: Pearson, 2011.

VARGA, A.; HORNIG, R. An overview of the OMNeT++ simulation environment. *SIMU-Tools 2008 - 1st International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems*, 2008.

VIANNA, A. S. G. *et al.* Análise do Impacto de Chuvas na Velocidade Média do Transporte Público Coletivo de Ônibus em Recife. In: *I Workshop Brasileiro de Cidades Inteligentes*. [S.l.: s.n.], 2018.

VILLARIM, M. R. *et al.* LoRa performance assessment in dense urban and forest areas for environmental monitoring. *INSCIT 2019 - 4th International Symposium on Instrumentation Systems, Circuits and Transducers*, 2019.

VILLARIM, M. R. *et al.* Calibration of a Low Cost Sensor for PM2.5 using a Reference PM Monitoring Station. *6th International Conference on Sensors Engineering and Electronics Instrumentation Advances (SEIA' 2020)*, 2020.

VIRTUALYS *et al.* *CupCarbon User Guide - Version U-One 4.1*. 2019. Disponível em: <http://pagesperso.univ-brest.fr/~bouceur/cupcarbon/doc/cupcarbon_user_guide.pdf>.

WANDER, A. S. *et al.* Energy Analysis for Public-Key Cryptography for Wireless Sensor Network. In: *Third IEEE International Conference on Pervasive Computing and Communications (PerCom)*. [S.l.]: IEEE, 2005. p. 324–328.

Wi-SUN Alliance. *Wi-SUN Alliance*. 2021. Disponível em: <<https://wi-sun.org/>>.

XLIM *et al.* *CupCarbon - A Smart City & IoT WSN Simulator*. 2020. Disponível em: <<http://cupcarbon.com/>>.

ZANELLA, A. *et al.* Internet of things for smart cities. *IEEE Internet of Things Journal*, IEEE, v. 1, n. 1, p. 22–32, 2014. ISSN 23274662.

ZANGENEH, M. A.; GHAZVINI, M. An energy-based clustering method for WSNs using artificial bee colony and genetic algorithm. *2nd Conference on Swarm Intelligence and Evolutionary Computation, CSIEC 2017 - Proceedings*, IEEE, p. 35–41, 2017.

Apêndices

A1 - Tabelas de Resultados de Simulação para PDR

Tabela 5: Resultados de simulação do protocolo DVR para métrica PDR.

Velocidade do nó fonte (km/h)	PDR (%)	Desvio padrão	Margem de Erro
9	93,2078%	$\pm 0,0962\%$	$\pm 0,0843\%$
18	63,4711%	$\pm 1,4430\%$	$\pm 1,2648\%$
36	52,7973%	$\pm 4,5394\%$	$\pm 3,9789\%$
54	53,0508%	$\pm 4,8154\%$	$\pm 4,2208\%$
72	46,3806%	$\pm 0,9382\%$	$\pm 0,8224\%$
90	23,6286%	$\pm 1,2063\%$	$\pm 1,0574\%$
108	35,9880%	$\pm 2,8676\%$	$\pm 2,5135\%$
126	29,8270%	$\pm 2,4679\%$	$\pm 2,1632\%$
144	23,3813%	$\pm 2,3422\%$	$\pm 2,0530\%$
162	9,1343%	$\pm 1,1081\%$	$\pm 0,9713\%$
180	28,0947%	$\pm 1,4667\%$	$\pm 1,2856\%$

Fonte: Elaborado pelo autor.

Tabela 6: Resultados de simulação do protocolo AODV para métrica PDR.

Velocidade do nó fonte (km/h)	PDR (%)	Desvio padrão	Margem de Erro
9	98,3193%	$\pm 0,8403\%$	$\pm 0,7366\%$
18	97,1429%	$\pm 0,4603\%$	$\pm 0,4034\%$
36	92,9412%	$\pm 0,4603\%$	$\pm 0,4034\%$
54	91,9328%	$\pm 0,4603\%$	$\pm 0,4034\%$
72	86,8908%	$\pm 1,1274\%$	$\pm 0,9882\%$
90	84,2017%	$\pm 0,7031\%$	$\pm 0,6163\%$
108	81,1765%	$\pm 0,4603\%$	$\pm 0,4034\%$
126	81,8487%	$\pm 2,1092\%$	$\pm 1,8488\%$
144	76,3025%	$\pm 0,7031\%$	$\pm 0,6163\%$
162	76,9748%	$\pm 0,4603\%$	$\pm 0,4034\%$
180	74,9580%	$\pm 0,7031\%$	$\pm 0,6163\%$

Fonte: Elaborado pelo autor.

Tabela 7: Resultados de simulação do protocolo DSR para métrica PDR.

Velocidade do nó fonte (km/h)	PDR (%)	Desvio padrão	Margem de Erro
9	97,9832%	$\pm 0,4603\%$	$\pm 0,4034\%$
18	96,4706%	$\pm 0,3758\%$	$\pm 0,3294\%$
36	92,2689%	$\pm 0,3758\%$	$\pm 0,3294\%$
54	88,2353%	$\pm 0,5942\%$	$\pm 0,5208\%$
72	83,8655%	$\pm 0,7031\%$	$\pm 0,6163\%$
90	81,8487%	$\pm 1,4062\%$	$\pm 1,2325\%$
108	77,4790%	$\pm 0,7031\%$	$\pm 0,6163\%$
126	76,8067%	$\pm 0,7516\%$	$\pm 0,6588\%$
144	73,4454%	$\pm 0,7516\%$	$\pm 0,6588\%$
162	71,5966%	$\pm 1,0957\%$	$\pm 0,9604\%$
180	67,7311%	$\pm 0,4603\%$	$\pm 0,4034\%$

Fonte: Elaborado pelo autor.

Tabela 8: Resultados de simulação do protocolo DSR modificado para métrica PDR.

Velocidade do nó fonte (km/h)	PDR (%)	Desvio padrão	Margem de Erro
9	97,3109%	$\pm 0,3758\%$	$\pm 0,3294\%$
18	95,7841%	$\pm 0,0195\%$	$\pm 0,0171\%$
36	92,4256%	$\pm 0,5743\%$	$\pm 0,5034\%$
54	86,8908%	$\pm 0,7516\%$	$\pm 0,6588\%$
72	82,9939%	$\pm 1,8554\%$	$\pm 1,6263\%$
90	80,8403%	$\pm 0,3758\%$	$\pm 0,3294\%$
108	77,8151%	$\pm 0,4603\%$	$\pm 0,4034\%$
126	75,7584%	$\pm 0,6597\%$	$\pm 0,5783\%$
144	73,1876%	$\pm 0,6741\%$	$\pm 0,5908\%$
162	71,8843%	$\pm 0,9954\%$	$\pm 0,8725\%$
180	68,9076%	$\pm 0,8403\%$	$\pm 0,7366\%$

Fonte: Elaborado pelo autor.

A2 - Tabelas de Resultados de Simulação para E2ED

Tabela 9: Resultados de simulação do protocolo DVR para métrica E2ED.

Velocidade do nó fonte (km/h)	E2ED (s)	Desvio padrão	Margem de Erro
9	0,173599	$\pm 0,015905$	$\pm 0,013941$
18	0,120345	$\pm 0,014060$	$\pm 0,012323$
36	0,162985	$\pm 0,028000$	$\pm 0,024543$
54	0,170459	$\pm 0,023241$	$\pm 0,020372$
72	0,153413	$\pm 0,035871$	$\pm 0,031442$
90	0,111610	$\pm 0,015444$	$\pm 0,013537$
108	0,132952	$\pm 0,021976$	$\pm 0,019262$
126	0,180736	$\pm 0,023820$	$\pm 0,020878$
144	0,171812	$\pm 0,052234$	$\pm 0,045784$
162	0,154774	$\pm 0,049716$	$\pm 0,043577$
180	0,149120	$\pm 0,027620$	$\pm 0,024209$

Fonte: Elaborado pelo autor.

Tabela 10: Resultados de simulação do protocolo AODV para métrica E2ED.

Velocidade do nó fonte (km/h)	E2ED (s)	Desvio padrão	Margem de Erro
9	0,244776	$\pm 0,016688$	$\pm 0,014628$
18	0,163592	$\pm 0,024411$	$\pm 0,021397$
36	0,174586	$\pm 0,019165$	$\pm 0,016798$
54	0,198759	$\pm 0,019087$	$\pm 0,016730$
72	0,195969	$\pm 0,014342$	$\pm 0,012571$
90	0,193568	$\pm 0,012246$	$\pm 0,010734$
108	0,188066	$\pm 0,028704$	$\pm 0,025160$
126	0,199976	$\pm 0,022321$	$\pm 0,019565$
144	0,212916	$\pm 0,030055$	$\pm 0,026344$
162	0,174312	$\pm 0,013638$	$\pm 0,011954$
180	0,185935	$\pm 0,019694$	$\pm 0,017263$

Fonte: Elaborado pelo autor.

Tabela 11: Resultados de simulação do protocolo DSR para métrica E2ED.

Velocidade do nó fonte (km/h)	E2ED (s)	Desvio padrão	Margem de Erro
9	0,279489	$\pm 0,054119$	$\pm 0,047437$
18	0,215153	$\pm 0,037526$	$\pm 0,032893$
36	0,218977	$\pm 0,015654$	$\pm 0,013721$
54	0,237122	$\pm 0,061300$	$\pm 0,053731$
72	0,202492	$\pm 0,020503$	$\pm 0,017971$
90	0,222069	$\pm 0,026182$	$\pm 0,022949$
108	0,197073	$\pm 0,021696$	$\pm 0,019017$
126	0,214651	$\pm 0,019647$	$\pm 0,017221$
144	0,202291	$\pm 0,011616$	$\pm 0,010182$
162	0,201451	$\pm 0,014765$	$\pm 0,012942$
180	0,239016	$\pm 0,028088$	$\pm 0,024620$

Fonte: Elaborado pelo autor.

Tabela 12: Resultados de simulação do protocolo DSR modificado para métrica E2ED.

Velocidade do nó fonte (km/h)	E2ED (s)	Desvio padrão	Margem de Erro
9	0,271940	$\pm 0,020572$	$\pm 0,018032$
18	0,204704	$\pm 0,012660$	$\pm 0,011097$
36	0,206965	$\pm 0,022458$	$\pm 0,019685$
54	0,223358	$\pm 0,031471$	$\pm 0,027585$
72	0,243558	$\pm 0,026808$	$\pm 0,023498$
90	0,218504	$\pm 0,012338$	$\pm 0,010815$
108	0,216243	$\pm 0,020157$	$\pm 0,017668$
126	0,218772	$\pm 0,026655$	$\pm 0,023364$
144	0,211575	$\pm 0,019242$	$\pm 0,016866$
162	0,238693	$\pm 0,030244$	$\pm 0,026509$
180	0,249024	$\pm 0,032396$	$\pm 0,028396$

Fonte: Elaborado pelo autor.

A3 - Tabelas de Resultados de Simulação para JIT

Tabela 13: Resultados de simulação do protocolo DVR para métrica JIT.

Velocidade do nó fonte (km/h)	Jitter médio (s)	Desvio padrão	Margem de Erro
9	0,001071	$\pm 0,000351$	$\pm 0,000308$
18	0,000479	$\pm 0,000010$	$\pm 0,000009$
36	0,000568	$\pm 0,000054$	$\pm 0,000047$
54	0,002020	$\pm 0,000061$	$\pm 0,000053$
72	0,000661	$\pm 0,000021$	$\pm 0,000018$
90	0,001782	$\pm 0,000683$	$\pm 0,000599$
108	0,002237	$\pm 0,001198$	$\pm 0,001050$
126	0,007015	$\pm 0,005786$	$\pm 0,005071$
144	0,005199	$\pm 0,002655$	$\pm 0,002327$
162	0,006506	$\pm 0,009401$	$\pm 0,008240$
180	0,004088	$\pm 0,004437$	$\pm 0,003889$

Fonte: Elaborado pelo autor.

Tabela 14: Resultados de simulação do protocolo AODV para métrica JIT.

Velocidade do nó fonte (km/h)	Jitter médio (s)	Desvio padrão	Margem de Erro
9	0,002211	$\pm 0,002342$	$\pm 0,002053$
18	0,000782	$\pm 0,000009$	$\pm 0,000008$
36	0,000463	$\pm 0,000003$	$\pm 0,000002$
54	0,001238	$\pm 0,000014$	$\pm 0,000012$
72	0,004732	$\pm 0,006665$	$\pm 0,005842$
90	0,000528	$\pm 0,000212$	$\pm 0,000186$
108	0,002545	$\pm 0,002828$	$\pm 0,002479$
126	0,000544	$\pm 0,000214$	$\pm 0,000187$
144	0,002865	$\pm 0,003011$	$\pm 0,002639$
162	0,000526	$\pm 0,000115$	$\pm 0,000101$
180	0,003437	$\pm 0,005429$	$\pm 0,004759$

Fonte: Elaborado pelo autor.

Tabela 15: Resultados de simulação do protocolo DSR para métrica JIT.

Velocidade do nó fonte (km/h)	Jitter médio (s)	Desvio padrão	Margem de Erro
9	0,001245	$\pm 0,000223$	$\pm 0,000195$
18	0,000864	$\pm 0,000010$	$\pm 0,000009$
36	0,000981	$\pm 0,001209$	$\pm 0,001060$
54	0,001613	$\pm 0,000272$	$\pm 0,000238$
72	0,002898	$\pm 0,001595$	$\pm 0,001398$
90	0,000474	$\pm 0,000011$	$\pm 0,000010$
108	0,001076	$\pm 0,000012$	$\pm 0,000011$
126	0,000630	$\pm 0,000288$	$\pm 0,000252$
144	0,003299	$\pm 0,003010$	$\pm 0,002639$
162	0,000544	$\pm 0,000011$	$\pm 0,000009$
180	0,001811	$\pm 0,000794$	$\pm 0,000696$

Fonte: Elaborado pelo autor.

Tabela 16: Resultados de simulação do protocolo DSR modificado para métrica JIT.

Velocidade do nó fonte (km/h)	Jitter médio (s)	Desvio padrão	Margem de Erro
9	0,002414	$\pm 0,003412$	$\pm 0,002990$
18	0,001390	$\pm 0,001103$	$\pm 0,000967$
36	0,000753	$\pm 0,000277$	$\pm 0,000243$
54	0,001580	$\pm 0,000012$	$\pm 0,000010$
72	0,005776	$\pm 0,003835$	$\pm 0,003361$
90	0,001066	$\pm 0,000009$	$\pm 0,000008$
108	0,001100	$\pm 0,000013$	$\pm 0,000012$
126	0,001008	$\pm 0,000280$	$\pm 0,000245$
144	0,002261	$\pm 0,000808$	$\pm 0,000708$
162	0,000939	$\pm 0,000366$	$\pm 0,000321$
180	0,001549	$\pm 0,000715$	$\pm 0,000627$

Fonte: Elaborado pelo autor.

A4 - Tabelas de Resultados de Simulação para THR

Tabela 17: Resultados de simulação do protocolo DVR para métrica THR.

Velocidade do nó fonte (km/h)	Throughput (pacotes/s)	Desvio padrão	Margem de Erro
9	0,031004	$\pm 0,000462$	$\pm 0,000405$
18	0,022416	$\pm 0,000426$	$\pm 0,000374$
36	0,017627	$\pm 0,001489$	$\pm 0,001305$
54	0,017893	$\pm 0,001034$	$\pm 0,000907$
72	0,016752	$\pm 0,000340$	$\pm 0,000298$
90	0,010243	$\pm 0,000622$	$\pm 0,000545$
108	0,012613	$\pm 0,000514$	$\pm 0,000451$
126	0,010930	$\pm 0,000646$	$\pm 0,000566$
144	0,009224	$\pm 0,000580$	$\pm 0,000508$
162	0,011221	$\pm 0,001863$	$\pm 0,001633$
180	0,009966	$\pm 0,000596$	$\pm 0,000522$

Fonte: Elaborado pelo autor.

Tabela 18: Resultados de simulação do protocolo AODV para métrica THR.

Velocidade do nó fonte (km/h)	Throughput (pacotes/s)	Desvio padrão	Margem de Erro
9	0,033038	$\pm 0,000283$	$\pm 0,000248$
18	0,032643	$\pm 0,000156$	$\pm 0,000137$
36	0,031244	$\pm 0,000155$	$\pm 0,000136$
54	0,030891	$\pm 0,000152$	$\pm 0,000133$
72	0,029198	$\pm 0,000381$	$\pm 0,000334$
90	0,028298	$\pm 0,000239$	$\pm 0,000210$
108	0,027512	$\pm 0,000152$	$\pm 0,000133$
126	0,027739	$\pm 0,000714$	$\pm 0,000626$
144	0,025654	$\pm 0,000235$	$\pm 0,000206$
162	0,025866	$\pm 0,000154$	$\pm 0,000135$
180	0,025314	$\pm 0,000119$	$\pm 0,000104$

Fonte: Elaborado pelo autor.

Tabela 19: Resultados de simulação do protocolo DSR para métrica THR.

Velocidade do nó fonte (km/h)	Throughput (pacotes/s)	Desvio padrão	Margem de Erro
9	0,032928	$\pm 0,000153$	$\pm 0,000134$
18	0,032417	$\pm 0,000126$	$\pm 0,000110$
36	0,031018	$\pm 0,000126$	$\pm 0,000110$
54	0,029650	$\pm 0,000200$	$\pm 0,000175$
72	0,028181	$\pm 0,000237$	$\pm 0,000207$
90	0,027505	$\pm 0,000475$	$\pm 0,000416$
108	0,026260	$\pm 0,000238$	$\pm 0,000209$
126	0,026030	$\pm 0,000254$	$\pm 0,000222$
144	0,024689	$\pm 0,000254$	$\pm 0,000223$
162	0,024060	$\pm 0,000371$	$\pm 0,000325$
180	0,022915	$\pm 0,000243$	$\pm 0,000213$

Fonte: Elaborado pelo autor.

Tabela 20: Resultados de simulação do protocolo DSR modificado para métrica THR.

Velocidade do nó fonte (km/h)	Throughput (pacotes/s)	Desvio padrão	Margem de Erro
9	0,032696	$\pm 0,000123$	$\pm 0,000108$
18	0,032187	$\pm 0,000006$	$\pm 0,000006$
36	0,031019	$\pm 0,000126$	$\pm 0,000111$
54	0,029198	$\pm 0,000253$	$\pm 0,000221$
72	0,027841	$\pm 0,000678$	$\pm 0,000594$
90	0,027165	$\pm 0,000130$	$\pm 0,000114$
108	0,026374	$\pm 0,000152$	$\pm 0,000133$
126	0,025632	$\pm 0,000201$	$\pm 0,000176$
144	0,024521	$\pm 0,000236$	$\pm 0,000207$
162	0,024117	$\pm 0,000382$	$\pm 0,000334$
180	0,023236	$\pm 0,000235$	$\pm 0,000206$

Fonte: Elaborado pelo autor.