UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

VICTOR EMANUEL FARIAS DA COSTA BORGES

AVALIAÇÃO DE PROTOCOLOS PARA A INTERNET DAS COISAS EM REDES VEICULARES

CAMPINA GRANDE - PB

2021

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Programa de Pós-Graduação em Ciência da Computação

Avaliação de Protocolos para a Internet das Coisas em Redes Veiculares

Victor Emanuel Farias da Costa Borges

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Engenharia de Software

Danilo Freire de Souza Santos
(Orientador)

Campina Grande, Paraíba, Brasil

B732a Borges, Victor Emanuel Farias da Costa.
Avaliação de Protocolos para a Internet das Coisas em Redes Veiculares / Victor Emanuel Farias da Costa Borges. – Campina Grande, 2022.
91 f. : il. color.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2021.
"Orientação: Prof. Dr. Danilo Freire de Souza Santos".
Referências.

1. Redes Veiculares. 2. Internet das Coisas. 3. Protocolos. I. Santos, Danilo Freire de Souza. II. Título.

CDU 004.7:621.398(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA MARIA ANTONIA DE SOUSA CRB 15/398

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO CIENCIAS DA COMPUTACAO
Rua Aprigio Veloso, 882, - Bairro Universitario, Campina Grande/PB, CEP 58429-900

FOLHA DE ASSINATURA PARA TESES E DISSERTAÇÕES

VICTOR EMANUEL FARIAS DA COSTA BORGES

AVALIAÇÃO DE PROTOCOLOS PARA A INTERNET DAS COISAS EM REDES VEICULARES

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação como pré-requisito para obtenção do título de Mestre em Ciência da Computação.

Aprovada em: 22/12/2021

Prof. Dr. DANILO FREIRE DE SOUZA SANTOS, Orientador, UFCG

Prof. Dr. DALTON CÉZANE GOMES VALADARES, Examinador Interno, IFPE

Prof. Dr. EDMAR CANDEIA GURJÃO, Examinador Externo, UFCG

Documento assinado eletronicamente por **DANILO FREIRE DE SOUZA SANTOS**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 22/12/2021, às 15:59, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Dalton Cézane Gomes Valadares**, **Usuário Externo**, em 22/12/2021, às 16:05, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **EDMAR CANDEIA GURJAO**, **PROFESSOR 3 GRAU**, em 22/12/2021, às 16:28, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **2038740** e o código CRC **AC9A5405**.

Resumo

As Redes Veiculares vêm se tornando mais populares a cada dia, sendo amplamente aplicadas no contexto de Internet dos Veículos (IoV), uma variação da Internet das Coisas (IoT) onde seus objetos são automóveis. Essas redes têm como objetivo prover comunicação entre dois ou mais veículos, ou entre um veículo e a infraestrutura da estrada, que por sua vez pode se comunicar com redes externas. Com isso, essas redes propõem trazer benefícios aos seus usuários através de aplicações, que podem ser relacionadas à segurança física dos passageiros, como evasão de acidentes automobilísticos, ou relacionadas a conforto e conveniência, como informações do trânsito em tempo real. Todavia, para que essa estrutura seja bem implementada, é necessário que alguns critérios sejam atendidos. Por se tratar de uma rede que precisa da recepção de informações quase que instantâneas, sobretudo em aplicações relacionadas à segurança física, elas necessitam de uma latência extremamente baixa, ao mesmo tempo que uma largura de banda consideravelmente alta. Além disso, é necessário que haja uma interoperabilidade entre os dispositivos IoV e serviços externos IoT, como em aplicações de entretenimento. Por isso, é necessário o estudo de soluções a nível de aplicação que potencializam estes serviços. Considerando esta problemática, este trabalho avalia e compara protocolos de aplicação específicos para Redes Veiculares com protocolos de propósito geral usados em sistemas de IoT, com a finalidade de identificar quais podem atender às necessidades de conectividade e interoperabilidade. Esta avaliação também investiga se estes protocolos são adequados para uso com sistemas de Computação móvel na Borda, considerando cenários específicos de Redes Veiculares, como o agrupamento de veículos em linha (*platoons*), além de uma simulação baseada na cidade de Campina Grande. Os resultados da avaliação mostram que o protocolo IoT de aplicação *Constrained Application Protocol* (CoAP) é viável para uso em Redes Veiculares, viabilizando um atraso de comunicação dentro da faixa esperada para essas redes, e favorecendo a interoperabilidade com serviços externos, como em uma aplicação de *streaming*.

Abstract

Vehicular Networks are becoming more popular every day, being widely applied in the context of the Internet of Vehicles (IoV), a variation of the Internet of Things (IoT) where its objects are automobiles. These networks are intended to provide communication between two or more vehicles, or between a vehicle and a road infrastructure, which can communicate with external networks. Thus, these networks propose to benefit their users through applications, which can be safety-related, such as collision avoidance, or non-safety-related, such as real-time traffic information. However, some rules must be met for this structure to be well implemented. As it is a network that needs the reception of information almost instantaneously, especially in safety-related applications, it needs extremely low latency and considerably high bandwidth. Furthermore, there needs to be interoperability between IoV devices and external IoT services, such as in infotainment applications. Therefore, it is necessary to study application-level solutions that enhance these services. Considering this problem, this work evaluates and compares specific application protocols for Vehicle Networks with general-purpose protocols used in IoT systems, to identify which ones can present positive results regarding connectivity and interoperability needs. This evaluation also investigates whether these protocols are suitable for use with mobile computing systems on the edge, considering specific scenarios for Vehicular Networks, such as the grouping of vehicle platooning, in addition to a simulation based on the city of Campina Grande. The evaluation results show that Constrained Application Protocol (CoAP) is viable for use in Vehicular Networks, enabling a communication delay within the expected range for these networks, and favoring interoperability with external services, such as in a streaming application.

Agradecimentos

Gostaria de agradecer primeiramente a Deus, por ter me guiado durante toda essa jornada, por ter me dado saúde em tempos difíceis de pandemia e por ter me dado coragem e sabedoria para encarar todos os desafios nestes anos. Depois, a minha mãe Shirley e meus avós Rosélia e Barbosa, por sempre acreditarem no meu potencial e proverem todo o suporte necessário durante toda minha vida de estudante e mestrando para que eu pudesse chegar até aqui.

Ao meu orientador, Danilo, pela paciência e toda a atenção e aprendizado transmitidos desde quando ingressei no programa de mestrado em agosto de 2019. Esse mesmo agradecimento também se estende ao professor Ângelo que me auxiliou durante toda a pesquisa e participou de inúmeras discussões que culminaram na conclusão deste trabalho.

A todos os colegas que fazem parte do grupo de pesquisadores do Future Connected Systems Group, em especial a Dalton, pela ajuda em diversos momentos e por transmitir seus conhecimentos como pesquisador mais experiente, e a Krzysztof, que me auxiliou bastante com a escrita do artigo que originou este trabalho.

Aos professores Hyggo, Rohit, Antão, Fábio e Leandro, com os quais cursei disciplinas durante o programa de mestrado que me auxiliaram bastante na produção desta pesquisa.

Aos meus amigos que me acompanharam durante toda a minha caminhada universitária, em especial Gabriel, Mattheus Brito e Lucas Fernandes, com os quais tive a oportunidade de reencontrar alguns deles no programa de mestrado. Obrigado por toda a ajuda e por todos os momentos de descontração.

Para finalizar, gostaria de agradecer o apoio financeiro do programa de bolsas da Coordenação de Aperfeiçoamento de Pessoal e de Nível Superior.

Conteúdo

Acrônimos

IoT - *Internet of Things (Internet das Coisas)*

IoV - *Internet of Vehicles (Internet dos Veículos)*

RSU - *Road Site Unit*

OBU - *On Board Unit*

VANET - *Vehicular Ad-Hoc Network*

MANET - *Mobile Ad-Hoc Network*

5G - *Quinta Geração de Redes Móveis*

V2V - *Vehicle-to-Vehicle*

AU - *Application Unit*

V2I - *Vehicle-to-Infrastructure*

V2B - *Vehicle-to-Broadband-Cloud*

V2X - *Vehicle-to-Everything*

4G - *Quarta Geração de Redes Móveis*

WAVE - *Wireless Access in Vehicular Environments*

IEEE - *Institute of Electrical and Eletronics Engineers*

DSRC - *Dedicated Short Range Communication*

WSMP - *Wave Short Message Protocol*

CALM - *Communications Access for Land Mobiles*

ISO - *International Standard Organization*

C2CNet - *Car-to-Car-Net*

C2C-CC - *Car-to-Car-Consortium*

NED - *Network Description Language*

OTCL - *Object Oriented Tool Command*

WSN - *Wireless Sensor Network*

ISP - *Internet Protocol Suite*

OSI - *Open System Interconnection*

BSM - *Basic Safety Message*

HTTP - *Hypertext Transfer Protocol*

CoAP - *Constrained Application Protocol*

MQTT - *Message Queuing Telemetry Transport*

TCP - *Transmission Control Protocol*

UDP - *User Datagram Protocol*

QoS - *Quality of Service (Qualidade do Serviço)*

VIS - *V2X Information Service*

ITS - *Intelligent Transport Systems*

mMTC - *Massive Machine Type Communication*

Lista de Figuras

Lista de Tabelas

Lista de Códigos Fonte

Capítulo 1

Introdução

Uma das principais tendências no mundo da tecnologia é a Internet das Coisas. Também conhecida como IoT (*Internet of Things*, em inglês), esta área de pesquisa tem como objetivo a transformação de objetos cotidianos em dispositivos eletrônicos e a interconexão digital destes. A intensa pesquisa nesta área nos últimos anos tem possibilitado a inserção de diversos objetos no mundo do IoT, desde *smartbands*, *smartwatches*, máquinas de café, geladeiras, máquinas de lavar, e até mesmo automóveis.

Este último item tem ganho grande atenção na última década, tendo a rede que os conecta inclusive uma nomenclatura à parte: Internet dos Veículos, ou IoV (*Internet of Vehicles*, em inglês). O IoV diz respeito à aplicação do mesmo conceito de Internet das Coisas, onde as "coisas" são veículos. A Internet dos Veículos provê comunicação entre veículos com a infraestrutura da estrada através dos RSUs (*Road Side Units*) ou com outros veículos através dos OBUs (*On Board Units*). Um dos seus principais objetivos é ajudar as pessoas a obterem informação do tráfego em tempo real. Segundo Kombate e Wanglina [46], estas informações podem tornar a viagem mais confortável e conveniente, e até mesmo ajudar o motorista a evitar acidentes de trânsito.

Nos dias de hoje, muitos fabricantes de automóveis estão investindo em IoV nos seus produtos, como *General Motors, BMW, Mercedes Benz, Nissan* e *Volkswagen* [75]. Na prática, estas empresas oferecem alguns serviços de segurança como informações de topologia das estradas e evasão de colisão, como também serviços relacionados à conveniência e conforto, tais como informação do tráfego em tempo real e aplicações de entretenimento. Estas aplicações estão se tornando mais frequentes a cada dia, aumentando sua popularidade entre

os motoristas.

Figura 1.1: Imagem ilustrativa de uma Rede Veicular.

Para implementar a interconexão entre veículos e a rede, a Internet dos Veículos faz uso das Redes Ad-Hoc Veiculares, também conhecidas como VANETs (*Vehicular Ad-Hoc Networks*, em inglês), ilustradas na Figura 1.1. Basicamente, uma VANET funciona de maneira semelhante a uma Rede Ad-Hoc Móvel (MANET - *Mobile Ad-Hoc Network*, em inglês), onde cada veículo é um nó da rede [84]. É importante ressaltar que por se tratar de uma rede extremamente volátil (visto que os veículos estão em movimento a todo momento) e que precisa de respostas instantâneas, especialmente nas aplicações relacionadas à segurança do motorista, as VANETs possuem requisitos de conectividade importantes. Para um bom funcionamento da rede, é necessário uma baixa latência, bem como uma largura de banda elevada e alta disponibilidade [12]. Isso se deve ao fato de muitas dessas aplicações precisarem trocar informações entre entidades quase que instantaneamente. Garantir estes requisitos pode ser crucial em momentos de decisão crítica, como no serviço de evasão de acidentes, por exemplo. Esta, por sua vez, é uma das funcionalidades essenciais nas Redes Veiculares, visto que isso pode ajudar a reduzir o elevado número de fatalidades em acidentes

automobilísticos ao redor do mundo, como apresentado no gráfico da Figura 1.2.

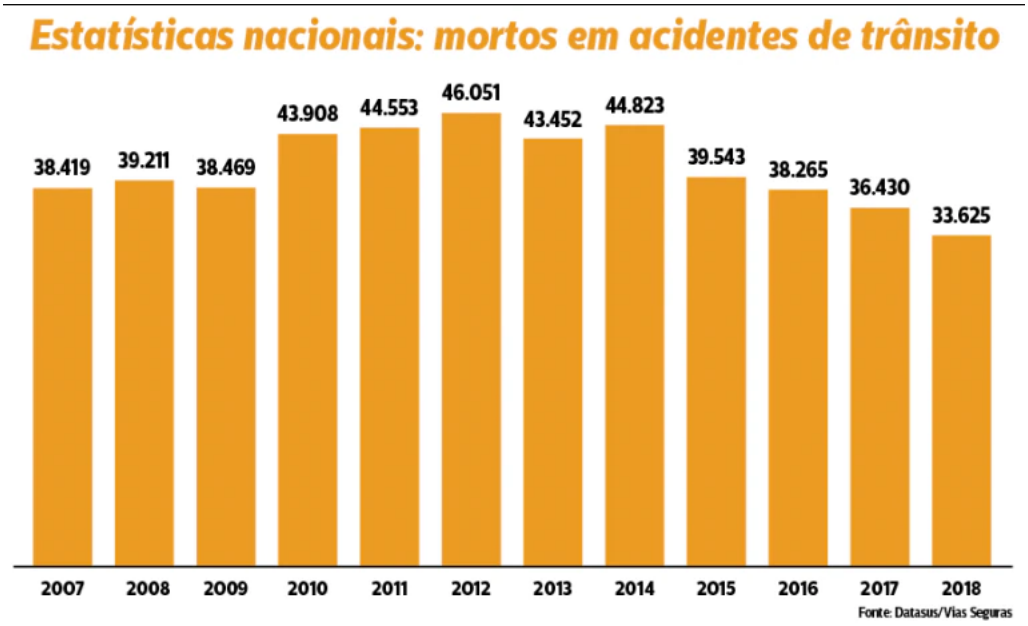

Figura 1.2: Número de fatalidades em acidentes de trânsito no Brasil entre os anos de 2007 à 2018 [22].

Além disso, é importante ressaltar que os pontos de troca de informações com a infraestrutura da estrada em algumas regiões podem se tornar um ponto de falha, ou seja, caso algum equipamento dos RSUs sofra um problema, a comunicação com redes externas torna-se impossível naquele momento [28]. Por fim, também é necessário que haja uma interoperabilidade entre os dispositivos IoV e outros serviços IoT, como por exemplo aplicações de clima e tráfego urbano. Para isso, é necessário o estudo de protocolos a nível de aplicação que potencializam estes serviços [12]. Um protocolo de aplicação é basicamente um conjunto de regras definidas para comunicação na camada de aplicação do modelo OSI [45]. No caso deste trabalho, serão considerados protocolos de aplicação voltados para soluções IoT, isto é, protocolos que são comumente utilizados para aplicações que envolvam as diversas áreas da Internet das Coisas. No entanto, não foram encontradas pesquisas envolvendo Redes Veiculares que investiguem aspectos de interoperabilidade com aplicações.

Portanto, as Redes Ad-Hoc Veiculares precisam atender a uma série de requisitos de conectividade para funcionarem de uma maneira satisfatória. Existe uma grande exigência com relação à latência, largura de banda e confiabilidade dos dados para garantir um bom funcionamento das aplicações de VANETs, sobretudo as relacionadas à segurança do motorista,

bem como importantes requisitos quanto à interoperabilidade. Por isso, precisa haver uma boa escolha de modelos de comunicação e protocolos de aplicação que sejam adequados neste panorama.

1.1 Objetivos do Trabalho

Tendo em vista o cenário exposto, o principal objetivo deste trabalho é investigar o impacto no desempenho de comunicação na Rede Veicular através da adoção de protocolos de comunicação a nível de aplicação utilizados em soluções para Internet das Coisas, tanto em termos de latência, como com relação a interoperabilidade entre serviços.

Outro importante objetivo deste trabalho é estudar se estes protocolos são viáveis para uso em um ambiente de Computação na Borda através da comunicação com redes móveis na topologia da VANET, como por exemplo redes 5G. Isto possibilita que a rede tenha uma melhor capacidade de largura de banda e promova maior segurança dos dados, bem como forneça uma estrutura que evite o problema de RSUs serem pontos únicos de falha na rede.

1.1.1 Questões de Pesquisa

Diante do contexto retratado, foram formuladas as seguintes questões de pesquisa para serem respondidas neste trabalho:

- *Questão de Pesquisa 1:* Quais protocolos de propósito geral adotados em IoT são vantajosos em Redes Veiculares quando comparados a soluções padrão?

  Um protocolo de propósito geral é vantajoso quando ele apresenta resultados melhores ou equivalentes em aplicações de Redes Veiculares quando comparado a soluções padrão, como um protocolo específico para VANETs. Sendo assim, deve-se descobrir quais apresentam um desempenho semelhante para cada aplicação avaliada, tanto considerando cenários de cidades reais quanto cenários que envolvem agrupamento de veículos em *platoons*.

- *Questão de Pesquisa 2:* Os protocolos para comunicação avaliados são adequados para um cenário de Computação na Borda em VANETs?

Nesta última questão, precisa-se avaliar se os protocolos que foram avaliados na questão de pesquisa anterior conseguem se adequar em um cenário em que a Computação na Borda se faz presente, com o objetivo de melhorar a conectividade da Rede Veicular.

1.1.2 Metodologia

Como metodologia, este trabalho caracteriza-se como uma experimentação, por conta da natureza causal das questões de pesquisa, tendo em vista o foco destas em observar o impacto da manipulação de variáveis, bem como por buscar uma investigação e proposta de estratégia de comunicação que envolva o uso de protocolos IoT em Redes Veiculares. O fluxo dos processos da metodologia adotada é apresentado na Figura 1.3.

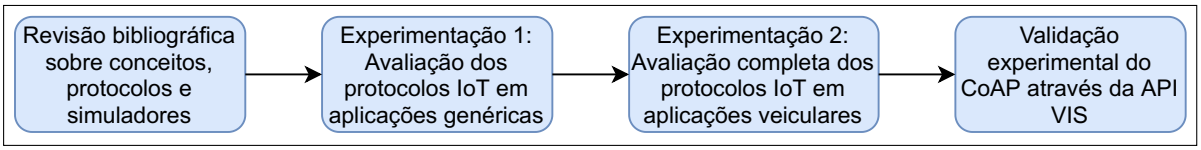

Figura 1.3: Fluxograma dos processos realizados no trabalho.

Inicialmente, foi realizada uma pesquisa de caráter bibliográfico acerca de todas as características relacionadas a Redes Veiculares, protocolos IoT e ferramentas de simulação que poderiam ser utilizadas na experimentação. Foram investigados os protocolos que melhor se encaixam no contexto de Redes Veiculares para que atendam todos os requisitos mencionados e favoreçam a interoperabilidade com serviços IoT.

Após a seleção dos protocolos e simuladores a serem utilizados, foi realizada a experimentação através de duas etapas: uma avaliação inicial dos protocolos IoT com aplicações genéricas, e uma avaliação completa dos protocolos em aplicações veiculares, ambas acerca de latência e largura de banda, com o objetivo de comparar o desempenho destes protocolos. Como cenários de estudo, as Redes Veiculares foram avaliadas através de ambientes que simulem desde cenários de circuitos fechados até bairros da cidade de Campina Grande, localizada no estado da Paraíba. Também foi avaliado um cenário que faz uso de uma estratégia de agrupamento de veículos em linha, também chamados de *platoons*. Cada *platoon* funciona como um grande nó na rede, tendo seu comportamento semelhante a um único nó no ponto de vista da topologia da VANET [68]. Neste contexto, vários veículos podem se

inserir em um *platoon* e guardam uma mesma distância dos automóveis vizinhos.

Na experimentação, foram comparados protocolos padrões de aplicações IoT que possuem uma boa adoção em cenários na Internet das Coisas, tais como CoAP e MQTT, com a solução e protocolo de comunicação padrão utilizado nas Redes Veiculares, conhecido como BSM. Para realizar o experimento, foi feito o uso das ferramentas *NetSim Standard*[1] e *Eclipse Mosaic*[2], dois simuladores integrados que possibilitam a experimentação e avaliação de parâmetros de rede em VANETs a partir de um determinado cenário, fazendo uso do *Sumo - Simulation of Urban Mobility*[3], um *software* de simulação envolvendo mobilidade urbana, que tem a funcionalidade de gerar o cenário veicular avaliado.

Por fim, o estudo contou com uma validação experimental através do desenvolvimento de um protótipo de serviço veicular fazendo uso da API *V2X Information Service*[4], um serviço facilitador para intermediação entre redes, com o propósito de avaliar o protocolo CoAP com relação à interoperabilidade. O objetivo desta validação é comprovar o uso do CoAP como estratégia de integração para facilitar a interoperabilidade na Rede Veicular por meio de um cenário envolvendo uma aplicação na borda da rede.

Todas estas ferramentas, bem como suas escolhas, serão mais detalhadas nos capítulos posteriores.

1.2 Relevância e Contribuições

Apesar de já ser conhecida como uma grande tendência nas montadoras internacionais de automóveis, a área de pesquisa relacionada à Internet dos Veículos ainda é recente quando comparada a outras linhas de pesquisa mais tradicionais na Ciência da Computação. Sendo assim, apesar de já existir um número considerável de trabalhos nesta área, ainda existem diversas lacunas a serem preenchidas. Além disso, ainda não existem Redes Veiculares implementadas no Brasil, principalmente por questões políticas e comerciais envolvendo o pouco espectro de radiofrequência disponível para o funcionamento destas tecnologias, problema que também explica a ausência das redes 5G no país [44].

[1] https://www.tetcos.com/netsim-std.html

[2] https://www.eclipse.org/mosaic/

[3] https://www.eclipse.org/sumo/

[4] https://forge.etsi.org/rep/mec/gs030-vis-api

Tendo em vista o contexto apresentado, os resultados deste trabalho devem contribuir para a melhoria de qualidade das VANETs disponíveis em países que já as possuem, por estudar aspectos de interoperabilidade que não são comumente explorados, além de atrair mais pesquisadores para a área de IoV. No cenário brasileiro, a pesquisa pode auxiliar a criação de Redes Veiculares no país em um futuro próximo, principalmente após a inclusão de redes 5G em nosso território, o que auxiliaria em sua implementação.

Além disso, é importante ressaltar que o estudo do uso de protocolos de aplicação IoT em Redes Veiculares visa ajudar a explorar questões quanto a interoperabilidade entre dispositivos e serviços IoV e IoT. Os resultados desta avaliação podem facilitar a integração dos veículos com serviços da Internet de propósitos gerais, abrindo um vasto leque de novas possibilidades de aplicações a serem implementados na Internet dos Veículos. Sendo assim, este trabalho tem um grande potencial de aplicabilidade de pesquisa e desenvolvimento.

Como contribuições alcançadas, este trabalho traz uma extensão para o simulador *Eclipse Mosaic* que permite a execução dos cenários de simulação com o protocolo CoAP, além de uma prova de conceito do serviço da API *V2X Information Service* implementada através de uma validação experimental.

Por fim, a introdução deste estudo contribuiu de forma a elencar os desafios de IoV com relação à conectividade, que foram citados anteriormente, e propor a avaliação que foi realizada neste trabalho, através de um artigo publicado na conferência *SoftCOM 2020* [12].

1.3 Organização do Trabalho

A presente dissertação está estruturada da seguinte forma. O Capítulo 2 apresenta uma revisão acerca de todos os conceitos e estado-da-arte em Redes Veiculares, abordando os principais tipos de comunicação, aplicações, padrões adotados para comunicação e o conceito de veículos em linha (*platoons*).

O Capítulo 3 apresenta uma análise comparativa das diversas ferramentas de simulação que podem ser utilizadas em um estudo de Redes Veiculares, bem como dos protocolos de aplicação que podem ser utilizados para este estudo.

O Capítulo 4 apresenta uma avaliação preliminar acerca de como estes protocolos de aplicação IoT de propósito geral se saem em testes utilizando um cenário artificial, através

de um mapa de uma cidade real, e ainda um cenário simulado de veículos organizados em *platoons* fazendo uso do *NetSim Standard*.

O Capítulo 5 é responsável por apresentar uma avaliação mais detalhada dos protocolos anteriores em aplicações em Redes Veiculares onde, fazendo uso de cenários realísticos e duas aplicações veiculares reais, os protocolos CoAP e BSM são testados e comparados fazendo uso do *Eclipse Mosaic*.

O Capítulo 6, por sua vez, finaliza o estudo através de uma validação experimental do protocolo de aplicação CoAP em Redes Veiculares com o objetivo de investigar se a adoção do CoAP é viável em termos de comunicação em um cenário envolvendo serviços externos na borda da rede, em uma implementação que faz uso da API *V2X Information Service* para facilitar a interoperabilidade.

Esta dissertação se encerra com o Capítulo 7, no qual são mostradas todas as conclusões obtidas com a realização deste trabalho, bem como a identificação de possíveis tópicos para trabalhos futuros.

Capítulo 2

Visão Geral sobre Redes Veiculares

O capítulo a seguir tem como objetivo apresentar os principais conceitos acerca das Redes Veiculares, desde o que são, os equipamentos de infraestrutura utilizados, arquiteturas, padrões de comunicação e aplicações veiculares, fornecendo assim uma visão geral de tudo o que foi trabalhado nesta pesquisa.

2.1 Definição

Com o recente avanço das tecnologias IoT e as evoluções das comunicações sem fio, vem surgindo um novo paradigma para garantir segurança veicular nas estradas, provendo comunicações entre veículos e a infraestrutura das vias, com o objetivo de uma troca de informações eficiente que traga mais segurança ao motorista e seus passageiros, bem como um maior conforto em sua viagem, economias de tempo e combustível, além de um melhor desempenho ao dirigir [61].

Neste paradigma, surgem as Redes Ad-hoc Veiculares (VANETs, do inglês *Vehicular Ad-hoc Networks*), que são um tipo de rede amplamente usada no campo de pesquisa de Internet dos Veículos. Por mais que algumas pessoas usem informalmente os conceitos de VANETs e IoV como sinônimos, eles não são a mesma coisa, visto que as Redes Ad-hoc Veiculares são uma forma de implementar a Internet dos Veículos. Sendo assim, todas as VANETs pertencem ao conceito de IoV, mas nem tudo que está englobado ao conceito de Internet dos Veículos possui uma interseção com VANETs [18].

As Redes Veiculares têm seu funcionamento de maneira bastante semelhante a uma Rede

Ad-hoc Móvel (MANET), com a diferença que os nós desta rede são caracterizados como veículos e pontos de infraestrutura de estrada, que tem como objetivo a troca de informações com redes externas.

Como qualquer rede de computadores, uma VANET possui uma estrutura bem definida pelas organizações responsáveis por padronizações de comunicação veicular [39]. Sendo assim, existem tipos e padrões de comunicação que são seguidos conforme sua pilha de protocolos.

As VANETs possui inúmeras aplicações, sejam relacionadas à segurança física dos passageiros, entretenimento, conforto e eficiência energética. Dentre as principais, destaca-se as funcionalidades de evasão de colisão, alertas de risco de acidentes e notificações de pós-colisão, uma vez que o objetivo principal destas redes é garantir que os motoristas tenham um tráfego seguro e, consequentemente, o número de acidentes automobilísticos, sobretudo os fatais, possa vir a diminuir.

Por fim, é importante ressaltar que as Redes Veiculares dependem de uma latência consideravelmente baixa para que consigam operar corretamente, com suas aplicações em funcionamento eficiente. Estudos indicam que o atraso desejável para aplicações veiculares não deve ser superior a 20 milisegundos, enquanto que o atraso máximo aceitável para o funcionamento de uma aplicação veicular se encontra em torno de 50 milisegundos [41], [15]. Sendo assim, é importante que as aplicações se adequem a estas faixas de latência, principalmente as que são relacionadas à segurança física dos passageiros, que se enquadram bem na faixa de atraso desejável.

2.2 Equipamentos de Infraestrutura

As Redes Veiculares convencionais precisam de equipamentos específicos em sua infraestrutura para funcionar corretamente. A literatura define dois domínios de infraestrutura categorizados: o domínio do veículo e o domínio da infraestrutura de estrada [48].

O domínio do veículo lida com todos os equipamentos necessários para que o automóvel esteja integrado à Rede Veicular. Sendo assim, o domínio é composto por um *On Board Unit* (OBU), um *hardware* que provê comunicação wireless do veículo com redes externas, e um ou mais *Application Units* (AUs), que são equipamentos auxiliares para aplicações. Estes,

por sua vez, são acoplados com o OBU, funcionando como um único dispositivo, como pode ser visto na Figura 2.1.

Figura 2.1: Foto de um *On Board Unit* da *V2XCast* [77].

Por outro lado, o domínio da infraestrutura de estrada tem como objetivo agrupar todos os equipamentos da VANET que estão no ambiente externo ao veículo, isto é, as ruas, avenidas e estradas. Este domínio é composto pelos *Road Side Units* (RSUs), que são antenas conectadas com veículos que suportam Redes Veiculares naquela região. Sua função é prover acesso à redes externas para os automóveis.

Do ponto de vista da topologia de rede, os OBUs podem ser vistos como os nós padrões de uma rede ad-hoc móvel, enquanto os RSUs são basicamente nós estáticos que estão conectados à redes externas.

2.3 Arquitetura de Comunicação

A junção de todos os equipamentos de infraestrutura mencionados na seção anterior permite com que os elementos que fazem parte da Rede Veicular possam se comunicar entre si. Essa comunicação, por sua vez, ocorre de forma estruturada em uma arquitetura.

A seguir, serão detalhados os principais tipos de comunicação presentes em uma Rede Ad-hoc Veicular [28]. A taxonomia dos tipos de comunicação pode ser vista na Figura 2.2.

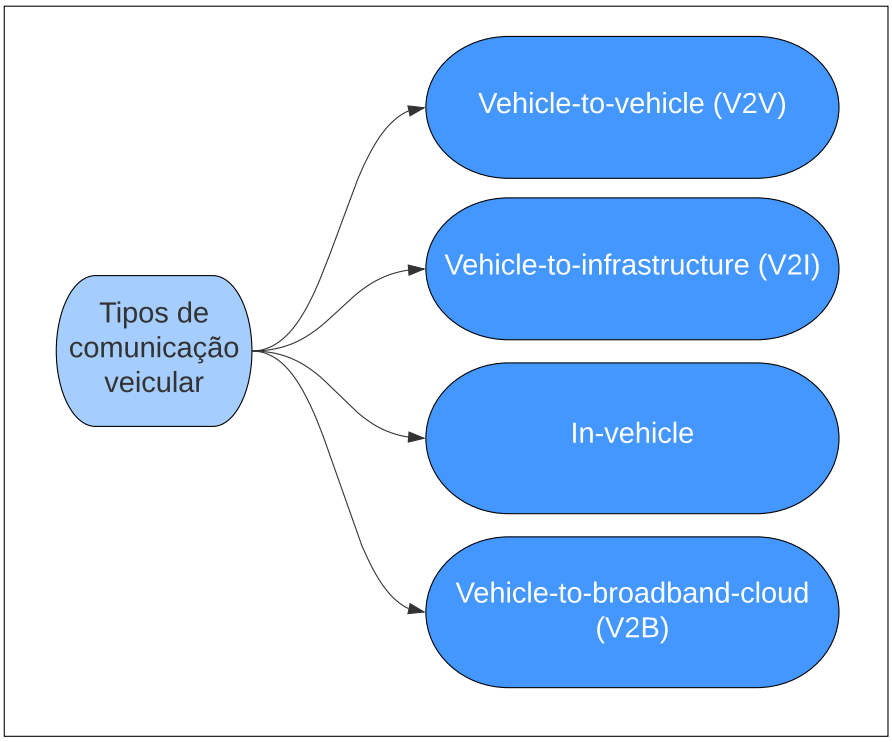

Figura 2.2: Taxonomia dos tipos de comunicação veicular.

2.3.1 Comunicação *Vehicle-to-vehicle* (V2V)

Sendo o tipo que atrai mais atenção dos pesquisadores, a comunicação *vehicle-to-vehicle* (V2V) provê a troca de dados entre veículos. Essa comunicação acontece entre os *On Board Units* que estão acoplados aos automóveis presentes na VANET.

Através da comunicação V2V, os veículos podem trocar informações relevantes para as aplicações veiculares, com o objetivo de proporcionar aos motoristas e passageiros uma viagem mais segura, bem como melhorar o desempenho do tráfego em grandes centros urbanos. O compartilhamento destes dados entre veículos podem informar detalhes sobre a velocidade na qual um dado veículo está trafegando, acelerações e desacelerações, além do posicionamento na estrada, mudanças no trajeto ou até mesmo interrupções no andamento da via como buracos e acidentes.

Em resumo, o tipo de comunicação V2V permite com que aplicações como assistência de direção, evasão de colisão ou notificação de pós-colisão possam funcionar de maneira precisa, como ilustrado na Figura 2.3(a).

2.3.2 Comunicação *Vehicle-to-infrastructure* (V2I)

Este tipo de comunicação provê a troca de informações e dados entre veículos e a infraestrutura da estrada. A comunicação ocorre entre os *On Board Units* dispostos nos veículos com acesso à VANET, e os *Road Side Units*, posicionados em determinadas posições da estrada percorrida.

Sendo assim, a comunicação *vehicle-to-infrastructure* (V2I) tem como objetivo permitir que haja um mecanismo de comunicação entre os veículos e a infraestrutura da Rede Veicular, permitindo assim auxiliar o controle do tráfego e o acesso à redes externas. No V2I, as informações enviadas pelos veículos podem auxiliar a infraestrutura na gestão das vias, possibilitando uma maior eficiência na atualização das condições de tráfego em determinados segmentos, bem como do trajeto que os veículos irão percorrer. Isso auxilia os motoristas a evitar engarrafamentos, selecionar caminhos mais eficientes e evitar problemas como buracos ou acidentes.

Além disso, os veículos podem solicitar acesso à redes externas através da comunicação V2I, permitindo assim o funcionamento de aplicações mais voltadas a servidores, tais como informações de tráfego em tempo real e atualizações de mudanças climáticas, como visto na Figura 2.3(b).

2.3.3 Comunicação *In-vehicle*

A comunicação *In-vehicle* refere-se à comunicação entre equipamentos do próprio automóvel, sendo assim um tipo de comunicação presente no domínio do veículo. Este tipo de comunicação permite aos sistemas veiculares analisarem a performance do veículo, podendo detectar problemas no funcionamento do mesmo.

Além disso, a comunicação *In-vehicle* também é capaz de detectar problemas do próprio motorista, como por exemplo aspectos de fadiga e cansaço caso seja identificada uma queda drástica no desempenho do mesmo na condução do automóvel.

2.3.4 Comunicação *Vehicle-to-broadband-cloud* (V2B)

Este tipo é relatado como a comunicação entre veículos e mecanismos de *broadband* como redes 4G e 5G. A *broadband cloud* pode incluir mais poder de transmissão dos dados,

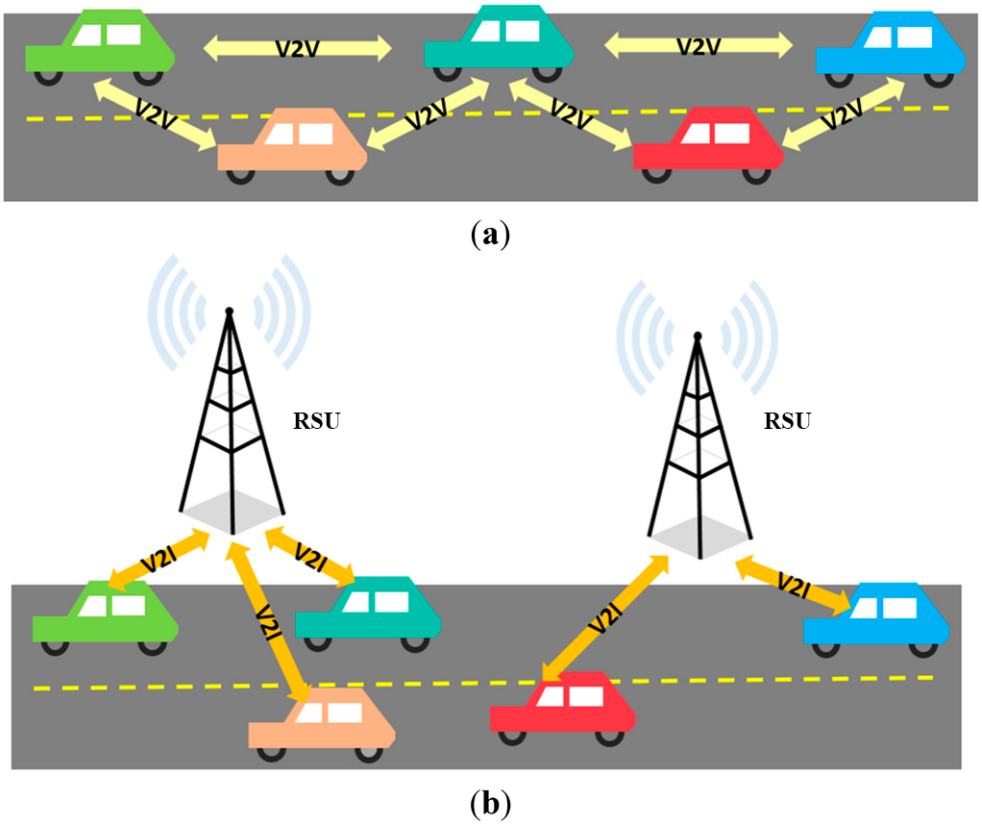

Figura 2.3: Esquema de comunicação *vehicle-to-vehicle* (a) e *vehicle-to-infrastructure* (b) [21].

sobretudo em aplicações de entretenimento.

Sendo assim, a comunicação *vehicle-to-broadband-cloud* visa inserir os veículos no universo das redes móveis de celular, permitindo assim uma integração com aplicações mais relacionadas à Internet das Coisas.

2.4 Padrões de Comunicação

Assim como qualquer tipo de rede de computadores, as Redes Veiculares possuem um padrão de comunicação bem definido, bem como seus protocolos que serão utilizados para implementar as comunicações, seja entre veículos, infraestrutura ou *broadband*.

Atualmente, existem três padrões de comunicação para VANETs, que serão detalhados a seguir [52].

2.4.1 WAVE - *Wireless Access in Vehicular Environments*

Desenvolvido pelo *Institute of Eletrical and Eletronics Engineers* (IEEE), o padrão *Wireless Access in Vehicular Environments* [18] é o mais utilizado em Redes Veiculares. Se caracterizando como uma junção dos padrões IEEE 802.11p e a família IEEE 1609, o WAVE implementa uma pilha de comunicação, que pode ser vista na Figura 2.4. Sua pilha se assemelha ao modelo OSI, onde há uma camada física implementada pelo padrão IEEE 802.11p, um multi-canal de operação em conjunto com o LLC atuando como camada de enlace, os protocolos IP, juntamente com o TCP e UDP, nas camadas de rede e transporte, podendo serem substituídos pelo protocolo WSMP (que será explicado a seguir), e por fim as aplicações veiculares, rodando sobre os padrões IEEE 1609.1 e IEEE 1609.2.

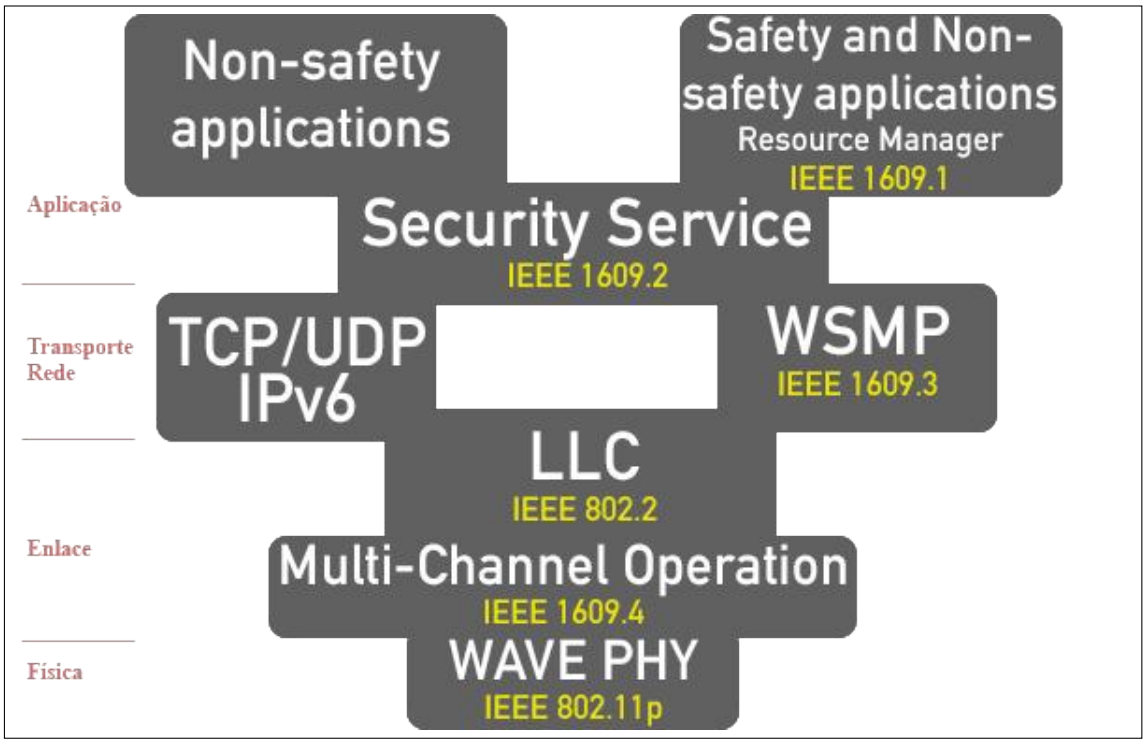

Figura 2.4: Pilha de protocolos WAVE para Redes Veiculares.

Como principais características, o padrão WAVE faz uso do *Dedicated Short Range Communication* (DSRC) em aplicações relacionadas à segurança, e também possui um protocolo de comunicação a nível de transporte próprio, chamado de *Wave Short Message Protocol* (WSMP) [83]. Este, por sua vez, pode substituir o uso de protocolos como IP, TCP e UDP.

Por ser o padrão de comunicação mais adotado, tanto em Redes Veiculares comerciais

como em estudos de caráter acadêmico, o padrão WAVE foi utilizado no estudo da presente dissertação [10].

2.4.2 CALM - *Communications Access for Land Mobiles*

O padrão *Communications Access for Land Mobiles* (CALM) [9] foi desenvolvido pela *International Standard Organization* (ISO) com o objetivo de criar uma estrutura que permitisse a comunicação de veículos e infraestrutura de estrada de forma contínua.

Dentre suas principais características, destaca-se a capacidade de prover comunicações cooperativas de forma heterogênea, suportando diversos tipos de tecnologia de comunicação, permitindo assim uma maior flexibilidade e adaptabilidade.

2.4.3 C2CNet - *Car-to-Car-Net*

Originado pela *Car-to-Car-Consortium* (C2C-CC), o padrão de comunicação *Car-to-Car-Net* (C2CNet) [8] é adotado em Redes Veiculares comerciais no território europeu e tem um grande foco sobretudo em aplicações relacionadas à segurança.

Sua principal característica está no fato de estruturar a Rede Veicular através de uma comunicação *multi-hop* baseada em roteamento geográfico, isto é, fazendo uso do posicionamento real dos veículos.

2.5 Aplicações Veiculares

Como foi visto nas seções anteriores, as Redes Veiculares possuem uma estrutura bem definida. No entanto, para que possua um real propósito ao usuário, as VANETs precisam de aplicações que façam uso da estrutura fornecida. Estas, por sua vez, são chamadas de aplicações veiculares, e possuem inúmeras possibilidades no ecossistema IoV, como mostrado na Figura 2.5.

A literatura classifica as aplicações veiculares em duas grandes categorias, que são vistas a seguir [35].

Figura 2.5: Alguns exemplos de aplicações IoV implementadas em Redes Veiculares.

2.5.1 Aplicações Relacionadas à Segurança

Estas aplicações têm como principal objetivo reduzir ou eliminar a probabilidade de acidentes automobilísticos. Sendo assim, o foco desta categoria é garantir a segurança física de todos os indivíduos envolvidos no contexto veicular, sejam motoristas, passageiros ou até pedestres.

Abaixo, serão detalhadas as principais aplicações veiculares relacionadas à segurança.

Aviso de colisão cooperativa (*Cooperative Collision Warning*)

Sendo a aplicação veicular mais relevante no contexto de VANETs, e também conhecida como evasão de colisão, tem como objetivo alertar aos veículos envolvidos que existe uma colisão iminente.

Esta é uma aplicação bastante complexa, visto que os veículos precisam avaliar o comportamento do motorista, bem como o movimento dos automóveis próximos, principalmente os que estão se movendo em sentido oposto. Ela alertará ao motorista se detectar perigo de colisão com outro veículo inserido na VANET, caso seja detectado [4].

Caso seja bem implementada, e a Rede Veicular garanta um bom desempenho em aspectos de conectividade, a aplicação para aviso de colisão cooperativa possui um grande

potencial de evitar futuros acidentes, e consequentemente salvar vidas no ambiente automobilístico.

Notificação de pós-colisão (*Post Crash Notification*)

Também conhecida como Aviso de interseção de colisão, esta é uma aplicação que não evita que haja um acidente de trânsito, mas mesmo assim pode garantir a segurança física dos envolvidos.

Após acontecer uma colisão que envolva um ou mais veículos inseridos na VANET, a aplicação automaticamente encaminha mensagens de *broadcast* sobre o ocorrido e a posição do acidente para os veículos próximos do local da colisão, para que eles possam evitar aquele lugar, bem como para ambulâncias e autoridades próximas ao local do acidente, solicitando suporte [33]. O funcionamento da aplicação é ilustrado na Figura 2.6.

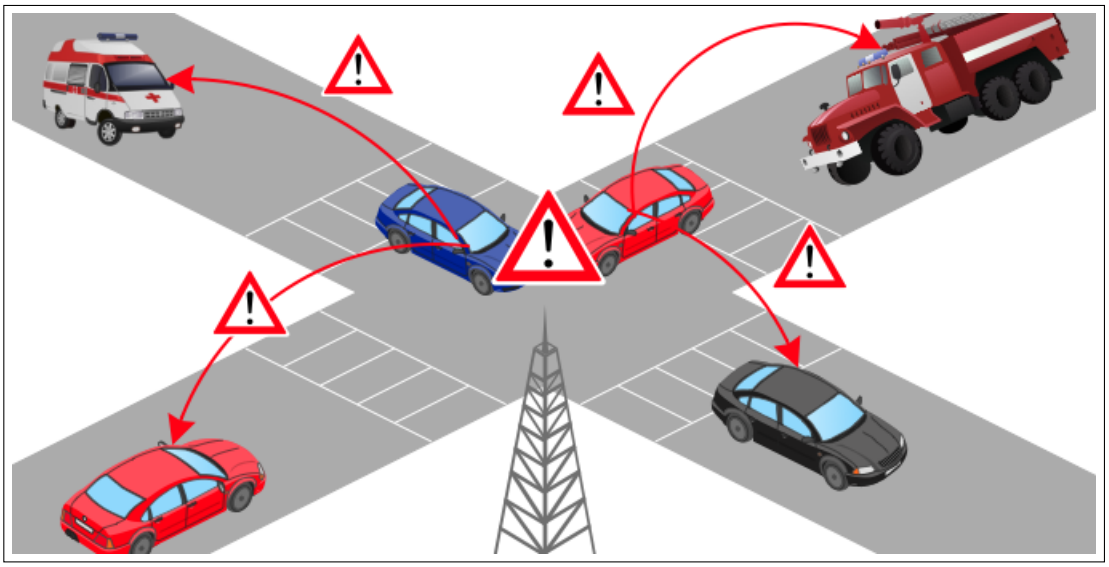

Figura 2.6: Imagem ilustrativa do funcionamento da aplicação de Notificação de pós-colisão.

Novamente, caso seja garantido um funcionamento ideal, a aplicação terá um grande potencial para evitar possíveis óbitos ou outras consequências no trânsito.

Alerta de mudança de faixa (*Lane Change Warning*)

Esta aplicação tem como objetivo alertar ao motorista que um ou mais veículos naquele segmento de estrada está fazendo uma mudança de faixa ou sentido, ou caso haja uma parada

brusca em automóveis próximos.

Alerta de aproximação de veículos de emergência (*Approaching Emergency Vehicle*)

A presente aplicação tem como foco avisar aos motoristas sobre a presença de veículos de emergência, tais como viaturas policiais, ambulâncias, caminhões do corpo de bombeiros, antes mesmo que eles se posicionem sobre o campo de visão do veículo. Sendo assim, o segmento de estrada pode ser desobstruído de maneira mais eficiente.

Aviso de curva acentuada (*Rollover Warning*)

Através desta aplicação, o motorista pode ser alertado que haverá uma curva muito acentuada a frente antes mesmo de chegar ao seu campo de visão. Sendo assim, o mesmo pode começar a reduzir sua velocidade antecipadamente.

Solicitação de serviços de ajuda (*Help Request*)

Nesta aplicação, o motorista pode solicitar serviços de ajuda caso haja algum problema em seu trajeto, como problemas mecânicos no automóvel. A aplicação irá encaminhar o exato posicionamento do veículo no segmento de estrada.

Alerta de zona em trabalho (*Workzone Warning*)

A aplicação tem como objetivo alertar aos veículos que uma zona em trabalho se aproxima, como por exemplo reparo em uma estrada. A notificação será recebida antes que aquela zona entre no campo de visão do veículo.

Alerta de ponto cego (*Blindspot Warning*)

Esta aplicação tem como foco diminuir e amenizar as situações em que o motorista tem sua visibilidade prejudicada por pontos que não são cobertos pelos espelhos retrovisores.

2.5.2 Aplicações não Relacionadas à Segurança

Estas aplicações são designadas para melhorar o trajeto do veículo, fornecendo conforto, entretenimento e conveniência ao motorista e passageiros. Sendo assim, o foco desta cate-

goria não está em garantir a segurança dos envolvidos, mas em aprimorar a viagem realizada com economias de tempo e combustível, bem como oferecendo serviços que tornem o trajeto mais confortável aos passageiros.

Abaixo, serão detalhadas as principais aplicações veiculares não relacionadas à segurança.

Notificação de pontos de interesse (*Interest Notification*)

Nesta aplicação, estabelecimentos comerciais como postos de gasolina, restaurantes, bares, lojas e outros tipos de serviço podem integrar-se à VANET. Sendo assim, o motorista ou os passageiros do veículo podem encontrar possíveis pontos de interesse de uma maneira mais eficiente.

Serviços de *streaming*

Assim como muitos cenários em IoT, as Redes Veiculares também podem ter serviços de *streaming* em sua estrutura. Estes podem ser relacionados à mídias de áudio, tais como *Spotify* e *Deezer*, ou mídias de vídeo, como *Netflix* e *YouTube*.

É importante ressaltar que, em caso de mídias de vídeo, alguns recursos podem ser desabilitados enquanto o veículo estiver em movimento para que o motorista não desvie sua atenção ao volante. No entanto, depende de onde estará acoplado o dispositivo que exibirá o conteúdo, pois se estiver em direção apenas aos passageiros, o serviço não terá essa limitação.

Informações de navegação (*Navigation Information*)

Aplicações que oferecem informações eficientes de navegação também podem ser inseridas no contexto de IoV. Estas aplicações normalmente serviços de localização via GPS em tempo real, e informa possíveis modificações no trânsito e trajeto.

As aplicações de navegação já são possíveis de serem usadas via smartphones, tais como *Google Maps* e *Waze*. Entretanto, a integração destas na Rede Veicular proporciona informações mais precisas, com um nível de detalhes muito maior quando comparado ao uso do aplicativo no dispositivo celular.

Informações de condições do clima (*Climate Change Information*)

Assim como o veículo pode ter acesso à informações de navegação, o mesmo também pode dispor de informações das condições do clima em segmentos de estrada que o motorista irá percorrer. A aplicação se mostra útil por permitir que todos os usuários possam verificar o clima de uma determinada região bem antes de chegar próximo àquela localidade.

Alerta de vagas de estacionamento

Com essa aplicação, o motorista pode ter acesso à informação de onde há vagas de estacionamento disponíveis naquela região. Esta é uma funcionalidade realmente útil, visto que evita com que o condutor perca muito tempo para encontrar uma vaga em determinada região, mas também é uma aplicação bastante complexa para ser implementada, já que as atualizações de vaga disponível ou indisponível precisam ser instantâneas.

Comboio de caminhões de carga (*Truck Platooning*)

Esta é uma aplicação veicular que também faz parte de outra sub-área do IoT chamada de *Smart Logistics* (Logística Inteligente). Nela, é possível direcionar um comboio de caminhões de uma mesma origem para um mesmo destino com a presença de apenas um motorista, que irá guiar o veículo líder, isto é, o caminhão mais a frente do comboio. A aplicação visa auxiliar a eficiência de entregas no setor de logística e faz uso da estratégia de agrupamento de veículos em linha (*vehicle platooning*), que será vista na seção a seguir.

2.6 Veículos em Linha (*Platoons*)

Como já foi mencionado na introdução deste trabalho, as Redes Veiculares são muito voláteis, por conta dos veículos estarem em constante movimento. Sendo assim, em determinados momentos e em determinadas regiões, os automóveis podem se desconectar dos *Road Side Units*, perdendo assim o acesso a redes externas.

O agrupamento de veículos em linha, conhecido como *vehicle platooning* em inglês, é uma estratégia adotada em algumas VANETs que agrupa vários veículos que estão se deslocando ao mesmo destino em um grande nó na rede, onde esses nós podem se comunicar

entre si. Essa estratégia visa tentar amenizar o problema da volatilidade, bem como melhorar a eficiência do tráfego e evitar sobrecarga da rodovia, como também reduzir a complexidade e aumentar a escalabilidade da topologia da rede [81].

O gerenciamento dos veículos em um *platoon* funciona da seguinte maneira. Um veículo que quer se inserir em um *platoon* transmite a mensagem para o líder do *platoon*. Este, por sua vez, tem a capacidade de aceitar ou rejeitar o veículo e, caso aceito, ele se insere no *platoon* e vai reduzindo sua velocidade gradativamente até se estabilizar, com a mesma velocidade dos outros veículos.

Quando um determinado veículo encontra seu destino, o mesmo pode deixar o *platoon* enviando uma mensagem ao líder. Este irá encaminhar uma mensagem a todos os veículos do *platoon* atrás daquele automóvel, para que reduzam sua velocidade, a fim de abrir espaço para que aquele veículo possa deixar o *platoon*. Após a saída deste, o *platoon* se restabelece com a mesma velocidade.

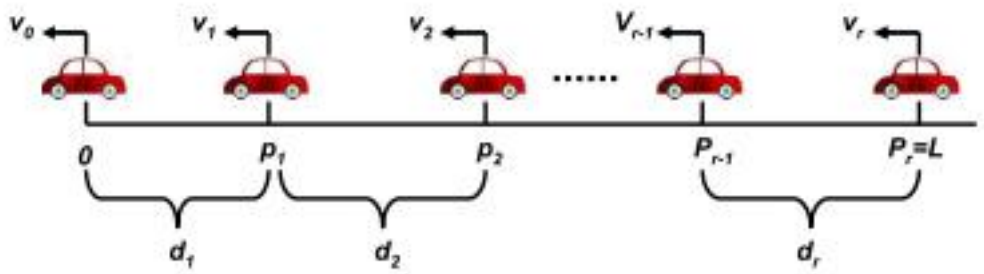

Figura 2.7: Imagem de um *platoon* de veículos [72].

Cada veículo inserido no *platoon* deve guardar uma distância igual e constante aos veículos vizinhos, ou seja, tanto o da sua frente, quando o de trás, como pode ser visto na Figura 2.7. É importante haver um certo cuidado ao guardar essa distância, pois erros no cálculo da velocidade dos veículos podem levar a colisões no mesmo *platoon*, fenômeno conhecido como *slinky effect* [36].

Esta estratégia foi usada em um dos cenários de avaliação do trabalho proposto.

2.7 Resumo do Capítulo

Como foi visto no decorrer do capítulo, as Redes Veiculares possuem equipamentos de infraestrutura específicos para cada domínio. Como muitos deles são inacessíveis e mais voltados ao âmbito comercial, foi optado por realizar a avaliação através de ferramentas de simulação, que foram levantadas no capítulo seguinte.

Com relação à arquitetura e padrões de comunicação, a pesquisa tem foco principalmente nas comunicações V2V e V2I, pela possibilidade de estudar a adoção de protocolos IoT nestas, e faz uso do padrão WAVE por ser predominantemente utilizado em pesquisas.

Também foi visto uma série de aplicações relacionadas e não relacionadas à segurança, onde algumas delas serão avaliadas nos Capítulos 5 e 6. Por fim, foi mostrada uma visão geral do conceito de *platoons* de veículos, que serão avaliados em cenários nos capítulos experimentais.

Capítulo 3

Análise dos Simuladores e Protocolos V2X

Neste capítulo, foi realizada uma análise acerca das ferramentas de simulação de Redes Veiculares, bem como dos protocolos de aplicação IoT, com o objetivo de selecionar quais destes fariam parte da avaliação experimental.

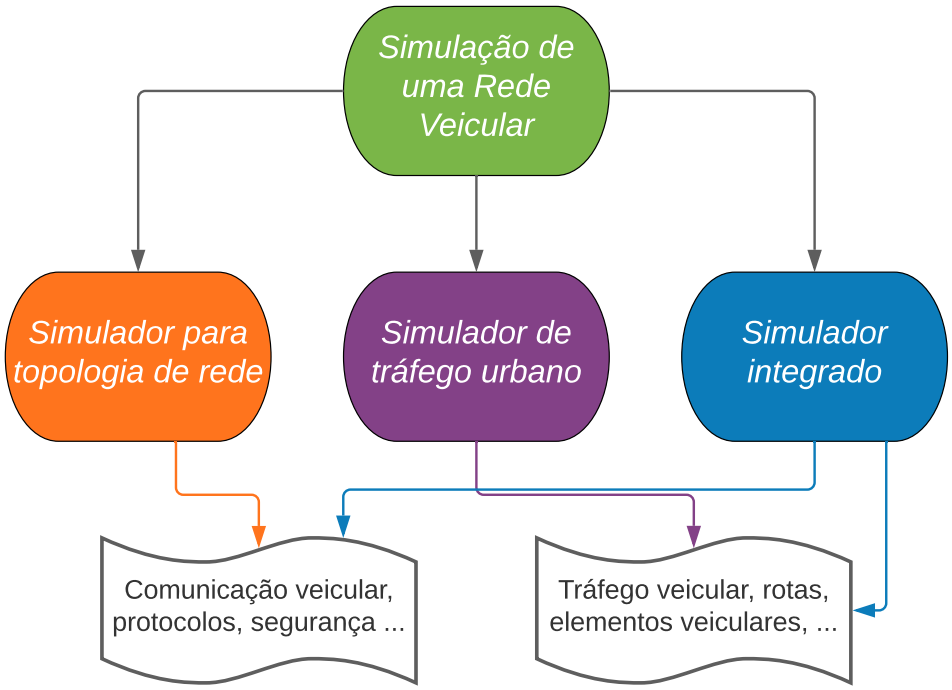

Figura 3.1: Componentes para a simulação de uma Rede Veicular.

Para realizar um estudo conciso, se faz necessário primeiramente avaliar todas as op-

ções de simuladores disponíveis para Redes Veiculares, para que seja possível escolher as melhores soluções, bem como a comparação e escolha dos protocolos de comunicação que mostram melhores resultados em soluções IoT de propósito geral.

A simulação de uma Rede Veicular precisa conter, como apresentado no diagrama da Figura 3.1, no mínimo duas componentes, isto é, dois tipos de simuladores com diferentes propósitos. O primeiro é um simulador para a topologia de rede, que lida com os aspectos técnicos da Rede Veicular dentro de suas camadas de comunicação. Já o outro é um simulador de tráfego urbano, que lida com a mobilidade dos veículos, a construção de rotas pelas quais irão transitar, e a topologia do ambiente simulado, incluindo semáforos, pedágios, pedestres, e todos os elementos de trânsito.

Além disso, também é possível ter um terceiro componente no ambiente de simulação, que é um *framework* de aplicação, também chamado de simulador integrado. Estes, por sua vez, integram ambos os simuladores citados anteriormente e adicionam novos recursos como aplicações veiculares. Sendo assim, o simulador integrado pode substituir o uso separado de um simulador para topologia de rede e um simulador de tráfego urbano.

Como neste estudo foram avaliados protocolos de aplicação, o uso deste componente foi fundamental na avaliação.

3.1 Simuladores da Topologia de Rede

Sendo um componente necessário para qualquer tipo de simulação relacionada à rede de computadores, os simuladores de rede são utilizados para realizar uma análise de como a rede implementada se comporta, desde a transmissão de dados entre nós, até os protocolos adotados em cada camada do modelo OSI. Com isso, faz-se possível identificar possíveis pontos que possam ser melhorados em sua topologia.

A seguir, são detalhados os principais simuladores da topologia de rede para VANETs.

3.1.1 OMNET++

O software OMNET++ [6] é um simulador de rede de código aberto amplamente utilizado por pesquisadores. Escrito na linguagem C++, destaca-se por permitir uma modelagem de redes de computadores com alta fidelidade, seja com ou sem fio.

Com sua estrutura modular, o OMNET++ permite a interligação dos módulos em conjuntos, formando uma composição destes. Para realizar essa interligação, o simulador faz uso do NED (*Network Description*), uma linguagem de descrição implementada para a própria ferramenta. As simulações realizadas podem ser executadas via linhas de comando ou por uma interface gráfica.

O OMNET++ possui dois *frameworks* de simulação na rede que podem ser utilizados, um de mobilidade que permite a construção de redes sem fio móveis, e o outro chamado *framework* INET, que possui um conjunto de módulos que permitem a representação de diferentes camadas dos protocolos da Internet, incluindo o padrão 802.11p utilizados pelas Redes Veiculares [56].

3.1.2 *Optimized Network Engineering Tools* - OPNET

Desenvolvido pela *OPNET Technologies*, o *Optimized Network Engineering Tools* [7] é um simulador de rede consolidado na área de engenharia por sua versatilidade e fidelidade. Atualmente, não é uma ferramenta de código aberto e é paga, apesar de possuir uma versão gratuita para uso acadêmico e não comercial chamada *OPNET IT Guru Academic Edition*.

O OPNET permite simular o comportamento de qualquer tipo de rede e seus dispositivos, tais como *switches*, *bridges*, servidores e roteadores, bem como protocolos e aplicações. Seus recursos permitem estudar os dispositivos de comunicação, determinados tipos de aplicação na rede, além de desempenhos de servidores ou protocolos utilizados no ambiente simulado.

Dentre os padrões de comunicação suportados pelo OPNET, há a presença dos padrões de comunicação sem fio, incluindo o IEEE 802.11, possibilitando o uso em Redes Veiculares [42]. No entanto, sua versão gratuita possui diversas limitações, tais como uma redução no tempo da simulação e o limite de entidades na rede.

3.1.3 NS-2 - *Network Simulator*

O *Network Simulator* 2 (NS-2) [57] é um simulador de rede totalmente gratuito e de código aberto. Escrito na linguagem C++ e OTCL (*Object Oriented Tool Command*, criada pelos desenvolvedores para ser usada no próprio simulador), possui uma grande variedade de

módulos já integrados, como componentes de protocolos de roteamento e aplicações, bem como módulos que são desenvolvidos pela comunidade que ainda suporta a ferramenta.

Após um arquivo de simulação ser executado, a saída é gerada por meio de arquivos de texto ou planilha através do *NetAnim* (*Network Animator*), que já é incluso no simulador e podendo ser utilizado para visualizar a simulação de uma VANET [65].

Já existe uma versão mais recente do *Network Simulator*, o NS-3, que será visto a seguir. Entretanto, como algumas modificações no funcionamento interno foram feitas, a maior parte dos códigos implementados para o NS-2 não funciona no NS-3. Isso proporcionou com que o NS-2 não fosse substituído, e ainda continua tendo suporte da comunidade.

3.1.4 NS-3 - *Network Simulator*

Construído com o objetivo de substituir a versão anterior, o *Network Simulator* 3 (NS-3) [58] é um simulador de rede gratuito e de código aberto. Diferentemente de seu antecessor, o NS-3 possui seu código inteiramente programado na linguagem C++, podendo também usar *Python* opcionalmente, ficando a cargo do usuário decidir. Por conta dessa alteração, a maior parte dos códigos implementados para o NS-2 não funcionam na versão nova, o que fez com que ambos originassem comunidades distintas.

Como novas funcionalidades que estavam ausentes no antecessor, o NS-3 possui suporte a nós com múltiplas interfaces de rede, endereçamento IPv6 e detalha melhor o padrão de comunicação 802.11, embora os padrões do NS-2 possam ser portados.

As saídas das simulações continuam a ser gerenciadas pelo *NetAnim*, que possui mais recursos, como gerar arquivos de *packet trace* para uma análise mais detalhada [16].

3.1.5 Comparação entre os Simuladores de Topologia de Rede

A Tabela 3.1 apresenta um resumo comparativo entre as principais características observadas nos simuladores avaliados. Nela, são avaliados os seguintes atributos:

- *Gratuito:* Avalia se o simulador é ou não sem custo. Versões gratuitas temporárias para teste (*trial*) não são válidas nesse quesito.
- *Código Aberto:* Diz respeito ao simulator ter seu código disponível para *download* e para edição.

Tabela 3.1: Comparação entre simuladores de topologia de rede.

	OMNET++	**OPNET**	**NS-2**	**NS-3**
Gratuito	Sim	Não	Sim	Sim
Código Aberto	Sim	Não	Sim	Sim
Interface Gráfica	Sim	Sim	Sim (parcial)	Sim (parcial)
Configuração	Médio	Médio	Médio	Médio
Uso	Médio	Médio	Fácil	Fácil
Recursos	Bom	Bom	Bom	Bom
Linguagem	C++	C++	C++ e OTCL	C++ ou Python
IEEE 802.11p	Sim	Sim	Sim	Sim

- *Interface Gráfica:* Corresponde ao fato da ferramenta ter ou não uma interface visual. Pode ser parcial, caso nem todos os recursos tenham suporte à interface gráfica.
- *Configuração:* Avalia a facilidade de configuração das ferramentas presentes no simulador.
- *Uso:* Avalia a facilidade de uso das funcionalidades presentes do simulador.
- *Recursos:* Avalia a quantidade e variedade de recursos que o simulador oferece com relação à Redes Veiculares. Caso ofereça uma ampla variedade de recursos úteis em simulações de VANETs, o simulador pode ser considerado "bom"neste atributo.
- *Linguagem:* Refere-se à linguagem de programação usada no simulador.
- *IEEE 802.11p:* Avalia a presença ou ausência dos padrão de comunicação IEEE 802.11p.

Nota-se que os simuladores *Omnet++*, *NS-2* e *NS-3* se destacam por serem gratuitos e de código aberto, facilitando assim o seu uso com relação a estudos acadêmicos. Além disso, ambos possuem uma boa quantidade de recursos e suporte da comunidade, além de não terem uma usabilidade complexa.

Por fim, todos os simuladores de rede avaliados possuem suporte ao padrão de comunicação IEEE 802.11p, possibilitando assim o uso em pesquisas de Redes Veiculares.

3.2 Simuladores de Tráfego Urbano

Outro componente importante para a simulação de uma Rede Veicular, os simuladores de tráfego urbano são responsáveis por realizar a simulação de um ambiente urbano, com todos os elementos que envolvem o tráfego de veículos.

Para simular este ambiente da maneira semelhante ao mundo real, os simuladores de tráfego fazem uso de um modelo de mobilidade, que são responsáveis por construir um ambiente realista semelhante a uma cidade, bem como definir as regras que cadenciam a maneira que os veículos se movimentam em suas ruas, avenidas e estradas. Com isso, é possível a simulação de cruzamentos, semáforos, delimitações de ruas e até mesmo faixas de pedestre.

A seguir, são detalhados os principais simuladores de tráfego urbano para Redes Veiculares.

3.2.1 MOVE

O MOVE [47] (*Mobility Model Generator for Vehicular Networks*) é um simulador de tráfego e mobilidade de código aberto que tem como objetivo facilitar a geração de modelos de mobilidade realísticos para simulações de Redes Veiculares. Escrito em *Java*, a ferramenta foi construída sobre o simulador SUMO, que ainda será visto nessa seção.

Como saída, o MOVE retorna um arquivo de rastreamento de mobilidade que possui informações de movimentos realistas dos veículos que compõem a VANET. Estas informações podem ser usadas por qualquer ferramenta de simulação que o suporte, sendo a principal o *Network Simulator* 2 (NS-2).

Além dessas características, o MOVE fornece um conjunto de interfaces gráficas ao usuário, que permitem gerar rapidamente cenários de simulação realistas sem o incômodo de precisar escrever *scripts* de simulação.

3.2.2 *CityMob*

O simulador *CityMob* [79] é um gerador de padrões de mobilidade urbana criado especialmente com o objetivo de estudar diferentes modelos de mobilidades em Redes Veiculares, bem como seu impacto no desempenho da comunicação entre veículos.

A ferramenta é capaz de criar cenários completos de mobilidade e simular carros que foram danificados em colisões, usando a rede para enviar informações a outros veículos, evitando assim acidentes automobilísticos ou engarrafamentos [51].

3.2.3 *VanetMobiSim*

O *VanetMobiSim* [27] surgiu como uma extensão ao *CanuMobiSim*, um simulador baseado em *Java* que permite a geração de modelos de mobilidade em diferentes formatos, suportando diferentes ferramentas de simulação de rede como NS-2, *GloMoSim*, entre outras.

A extensão *VanetMobiSim* diferencia-se por se concentrar na mobilidade veicular, sendo assim aplicada à Redes Veiculares. Com isso, ela apresenta modelos de movimento automotivo realistas em níveis macroscópicos e microscópicos, suportando mapas da base TIGER, bem como também podendo gerá-las aleatoriamente [37].

De acordo com estes modelos, os veículos podem regular sua velocidade dependendo dos carros próximos, ultrapassam outros e agem de acordo com os semáforos e placas na presença de cruzamentos.

3.2.4 SUMO

Desenvolvido pela *German Aerospace Center* (DLR), o SUMO (*Simulation for Urban Mobility*) [14] é o principal simulador de tráfego e mobilidade para o estudo de Redes Veiculares, sendo o mais utilizado na maioria dos estudos realizados nessa área. Por ser uma ferramenta de código aberto, possui um grande suporte, tanto dos desenvolvedores quanto da comunidade que o utiliza.

Como uma de suas suas principais características, o SUMO permite a geração de modelos de mobilidade através da ferramenta *osmWebWizard*, que faz uso do *OpenStreetMaps* para obter o mapa da região selecionada. Sendo assim, essa ferramenta facilita a criação de um cenário realista, onde o usuário pode personalizar como ocorrerá o movimento dos veículos [64].

O SUMO também tem como característica a individualização de cada veículo, coletando informações como o tipo do automóvel, sua atual posição, velocidade, faixa que está percor-

Tabela 3.2: Comparação entre simuladores de tráfego urbano.

	MOVE	**CityMob**	**VanetMobiSim**	**SUMO**
Gratuito	Sim	Sim	Sim	Sim
Código Aberto	Sim	Sim	Sim	Sim
Interface Gráfica	Sim	Sim	Sim	Sim
Configuração	Fácil	Médio	Médio	Fácil
Uso	Médio	Médio	Médio	Médio
Recursos	Bom	Bom	Médio	Bom
Linguagem	Java	C	Java	C++
Simuladores suportados	NS-2, NS-3, Omnet++	NS-2, NS-3	NS-2, NS-3, Omnet++	NS-2, NS-3, Omnet++

rendo, dentre outras. Ele também permite importar mapas disponíveis na Internet por meio da ferramenta "*Netconvert*".

3.2.5 Comparação entre os Simuladores de Tráfego Urbano

A Tabela 3.2 apresenta um resumo comparativo entre as principais características observadas nos simuladores avaliados. Os atributos apresentados são basicamente os mesmos da tabela anterior, com a seguinte adição:

- *Simuladores suportados:* Diz respeito aos simuladores de topologia de rede que são aceitos por cada simulador de tráfego urbano.

É possível notar que o simulador SUMO é o mais adotado nas pesquisas envolvendo tráfego veicular. Suas características favorecem sua escolha, como ser uma ferramenta de código aberto, facilidade de configuração, além de uma boa quantidade de recursos e simuladores de rede suportados.

Com relação à geração de modelos de mobilidade, os simuladores MOVE e *CityMob* também se destacam, sendo este último uma escolha viável para pesquisas que utilizam exclusivamente a linguagem *Java*.

3.3 Simuladores Integrados

Com a integração de um simulador da topologia de rede com um simulador de tráfego urbano, já é possível realizar simulações com Redes Veiculares. No entanto, caso seja feito

o uso de apenas essas duas ferramentas, a simulação será realizada apenas nas camadas de rede e transporte, sendo adequado para pesquisas sobre protocolos de roteamento.

Como o intuito desta pesquisa está na camada de aplicação, visto que é necessário simular aplicações veiculares para avaliar os protocolos, foi feito o uso de um *framework* de aplicação, também conhecido como simulador integrado. Estes integram ambos os simuladores com o objetivo de prover suporte à simulação com aplicações relacionadas ou não relacionadas à segurança.

A seguir são apresentados os principais simuladores integrados utilizados pelos pesquisadores da área.

3.3.1 *VSimRTI*

Desenvolvida pelo *Daimler Center for Automotive Information Technology Innovations* (DCAITI), criado em 2006 por uma iniciativa conjunta da *Daimler AG* e *Technische Universitat Berlin*, a ferramenta *VSimRTI* (*V2X Simulation Runtime Infrastructure*) [29] oferece uma estrutura compreensível para a avaliação de soluções em sistemas de transporte inteligentes (ITS).

Escrito na linguagem Java, o *framework* destaca-se pela sua flexibilidade na área de Redes Veiculares, sobretudo por permitir uma simulação dinâmica de aplicações de mobilidade inteligente, tais como soluções de monitoramento e transmissão de imagens, sendo possível de avaliar seus impactos e benefícios. Também se destaca por combinar várias ferramentas *built-in* que são oferecidos ao pesquisador, tais como simuladores de condições climáticas e energia, permitindo que a simulação seja o mais realista possível e consiga abranger todos os cenários possíveis [63].

O *VSimRTI* tem como limitações funcionar apenas em plataformas *Unix*, bem como possuir uma estrutura de aplicação baseada em interações de classes, dificultando que protocolos de aplicação sejam adaptados para o mesmo.

3.3.2 TraNS

O simulador *TraNS* (*Traffic and Network Simulation Environment*) [59] foi desenvolvido pela *École Polytechnique Fédérale de Lausanne* (EPFL) e possui uma estrutura simples de

ser manuseada. Escrito em *Java* e C++, destaca-se por oferecer aplicações específicas para simulação de eventos em estradas, tais como acidentes ou interdição da rua. Possui uma grande escalabilidade e funciona em todos os sistemas operacionais [62].

Como limitações, o *TraNS* tem a obrigatoriedade de fazer uso do Sumo como simulador de mobilidade e do NS-2 como simulador de rede, não permitindo uma maior flexibilidade dentre as demais opções.

3.3.3 NCTUns

Desenvolvido para a simulação de redes de comunicação em um contexto geral, o NCTUns (*National Chiao University Network Simulator*) [82] é uma ferramenta de alta fidelidade na simulação de redes. A partir da versão 5.0, foi atualizado para suportar o padrão de comunicação IEEE 802.11p, permitindo assim o estudo de Redes Veiculares.

Sendo escrito na linguagem C++, o NCTUs tem como vantagens possuir uma interface gráfica intuitiva, que permite com que o pesquisador possa implementar redes e protocolos de maneira fácil, e avaliar os resultados a partir de animações gráficas [19].

No entanto, a principal limitação do mesmo é o fato de funcionar apenas em plataformas *Fedora*. Além disso, sua mais recente versão 6.0, chamada de *EstiNet*, é comercial e suporta apenas o sistema operacional *Red Hat Fedora* 11.

3.3.4 *GrooveNet*

Sendo o mais antigo dos *frameworks* avaliados, o *GrooveNet* [50] foi criado em 2006 e possui um simulador integrado que provê o estudo de Redes Veiculares. Como principais características, se destaca por permitir comunicação de veículos simulados e reais, bem como permitir com que o pesquisador importe mapas de uma base de dados chamada TIGER, que é disponível livremente. Além disso, possui uma boa escalabilidade, suportando milhares de veículos na rede.

Como limitações, destaca-se a idade do simulador, que é antigo e não recebe suporte recente, além de possuir uma quantidade muito pequena de materiais de apoio, o que dificulta estudos com a ferramenta.

3.3.5 *Ventos*

Criado pela *University of California*, o *Ventos* (*Vehicular Network Open Simulator*) [78] é um simulador escrito em C++ integrado com o objetivo de estudar fluxos de tráfego veicular, direção colaborativa e interações entre veículos e infraestrutura. Como características relevantes, o *Ventos* possui suporte total ao padrão de comunicação DSRC, além de suporte para a simulação nativa de platoons e seus protocolos de gerenciamento [11].

A principal limitação do Ventos está no fato de ser uma ferramenta muito recente, que ainda não foi amplamente usada por pesquisadores e não possui muito suporte à comunidade.

3.3.6 *Veins*

O *Veins* (*Vehicles in Network Simulation*) [71] é um framework de simulação de Redes Veiculares de código aberto escrito em C++. Como seus destaques, a ferramenta possui a possibilidade de reconfiguração e roteamento online de veículos, além de suportar os modelos completos do IEEE 802.11p com WAVE e DSRC. Também permite que o pesquisador trabalhe com recursos *third-party* tais como módulos de rede celular.

Como limitações, o *Veins* possui uma difícil instalação e uso, além de não possuir suporte nativo à aplicações veiculares.

3.3.7 *NetSim*

Desenvolvido pela empresa *Tetcos*, o *NetSim* (*Network Simulator and Emulator*) [74] é um simulador integrado que possui sua própria estrutura de rede e pode fazer uso de qualquer simulador de tráfego. Escrito em C++, sua estrutura se destaca por ter uma interface gráfica intuitiva, além de ter suporte a uma vasta gama de tecnologias, dentre as principais IoT, VANETs, *Wireless Sensor Network* (WSN), *Software Defined Networks* (SDN) e redes 5G NR *mmWave*.

Para este trabalho, a característica mais importante a se destacar dessa ferramenta é a estrutura bem validada em sua camada de aplicação em Redes Veiculares, permitindo assim que quaisquer protocolos de aplicação possam ser implementados em aplicações veiculares.

Como limitações destaca-se o fato da ferramenta funcionar apenas no sistema operacional *Windows*, além de ser um simulador pago e não ser de código aberto.

3.3.8 *Eclipse Mosaic*

Desenvolvido pelo *Daimler Center for Automotive Information Technology Innovations* (DCAITI), o mesmo time responsável pelo *VSimRTI*, juntamente com a *Eclipse Foundation*, o simulador *Eclipse Mosaic* [32] é a mais recente das ferramentas avaliadas, sendo originado em outubro de 2020.

Escrito na linguagem C, a ferramenta, que é descrita como uma estrutura de simulação multi-domínio e multi-escala para mobilidade conectada e automatizada, nasceu a partir de uma compra do *VSimRTI* pela *Eclipse Foundation*. Sendo assim, a flexibilidade de combinar uma gama de ferramentas *built-in*, tais como Sumo, Omnet++, NS-3, SNS e outros, é mantida do antigo simulador, no entanto, novas funcionalidades foram adicionadas tais como acoplamento com outros tipos de simulador, possibilidade de criar modelos de *delay* e escalonamento de eventos. Além disso, outra adição importante foi a mudança da ferramenta de paga para código aberto, com uma versão monetizada mais completa chamada *Mosaic Extended*.

O *Eclipse Mosaic* tem basicamente as mesmas limitações do *VSimRTI*, tais como funcionar apenas em plataformas *Unix*, bem como possuir uma estrutura de aplicação baseada em interações de classes, dificultando que protocolos de aplicação sejam adaptados para o mesmo.

3.3.9 Comparação entre Simuladores Integrados

As Tabelas 3.3 e 3.4 apresentam um resumo comparativo entre as principais características observadas nos simuladores avaliados, aos quais foram divididos em duas tabelas por conta do grande número de ferramentas comparadas. Os atributos apresentados são semelhantes aos das tabelas anteriores, com as seguintes adições:

- *Suporta aplicações?:* Diz respeito ao suporte nativo ou não de aplicações veiculares.
- *Suporta protocolos de aplicação?:* Avalia se o simulador suporta nativamente protocolos de aplicação IoT.

Nota-se que apenas os simuladores integrados *VSimRTI*, *TraNS*, *NCTUns*, *NetSim* e *Mosaic* suportam aplicações, sendo assim apenas estes são úteis com relação ao que foi avaliado

Tabela 3.3: Comparação entre simuladores integrados (parte 1).

	VSimRTI	**TraNS**	**NCTUns**	**GrooveNet**
Gratuito	Sim	Sim	Não	Sim
Código Aberto	Sim	Sim	Não	Sim
Interface Gráfica	Não	Sim	Sim	Sim
Facilidade de Configuração	Médio	Médio	Difícil	Médio
Facilidade de Uso	Fácil	Médio	Fácil	Difícil
Recursos	Bom	Médio	Bom	Médio
Linguagem de Programação	Java / C	Java	C++	C++
Usa o padrão IEEE 802.11p?	Sim	Sim	Sim	Sim
Suporta aplicações?	Sim	Sim	Sim	Não
Suporta protocolos de aplicação?	Não	Não	Não	Não

nesta pesquisa. Dentre eles, destaca-se o simulador *NetSim*, que possui suporte nativo à protocolos de aplicação e, por este motivo, uma versão *trial* do mesmo foi utilizado em um dos experimentos desta pesquisa que serão vistos a seguir.

Além disso, o simulador *Eclipse Mosaic* também se destaca, principalmente por sua quantidade de recursos e por conter aplicações nativas que podem ser utilizadas pelo usuário, sendo necessário apenas a adaptação para protocolos de aplicação no mesmo. Por este motivo, o *Mosaic* também foi escolhido para ser utilizado nesta pesquisa.

3.4 Protocolos de Aplicação

Após selecionar as ferramentas que serão usadas para a simulação, é necessário estudar quais protocolos de aplicação são mais viáveis e que podem apresentar bons resultados no decorrer do trabalho.

Um protocolo de aplicação tem um objetivo semelhante a outros tipos de protocolos, isto é, coordenar e gerenciar a comunicação entre dois ou mais nós na rede. No entanto, seu foco está na camada de rede referente a aplicações. A camada de aplicação da rede é uma abstração que especifica as formas de comunicação e métodos de interface que serão utilizados por hospedeiros em uma rede de computadores [17].

Tabela 3.4: Comparação entre simuladores integrados (parte 2).

	Ventos	**Veins**	**NetSim**	**Mosaic**
Gratuito	Sim	Sim	Não	Sim
Código Aberto	Sim	Sim	Não	Sim
Interface Gráfica	Não	Não	Sim	Não
Facilidade de Configuração	Difícil	Difícil	Médio	Médio
Facilidade de Uso	Difícil	Difícil	Fácil	Fácil
Recursos	Bom	Médio	Bom	Bom
Linguagem de Programação	C++	C++	C++	Java / C
Usa o padrão IEEE 802.11p?	Sim	Sim	Sim	Sim
Suporta aplicações?	Não	Não	Sim	Sim
Suporta protocolos de aplicação?	Não	Não	Sim	Não

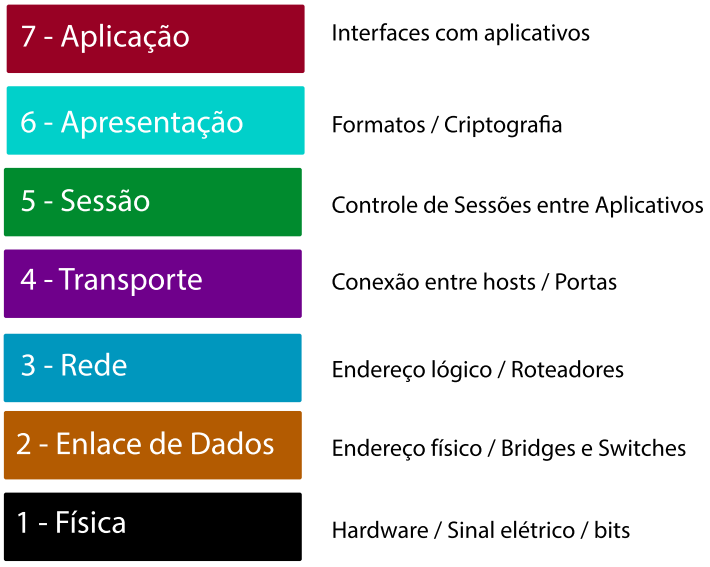

Figura 3.2: Camadas de rede segundo o modelo OSI [45].

Essa camada, por sua vez, está disponível em qualquer modelo em redes de computadores, seja o *Internet Protocol Suite* (ISP) [2] ou o *Open System Interconnection* (OSI) [70], que pode ser visto no diagrama da Figura 3.2. Sendo assim, é na camada de aplicação que ocorre a troca de mensagens fim a fim em aplicações de hosts distintos.

A seguir, serão apresentados os principais protocolos de aplicação que serão avaliados e

comparados no decorrer do trabalho.

3.4.1 *Basic Safety Message* - BSM

Basic Safety Message (BSM) é um padrão de comunicação que funciona de forma semelhante a um protocolo a nível de aplicação, utilizado exclusivamente para Redes Veiculares. Sua estrutura de mensagem é simples e leve, ao mesmo tempo que é orientada à conexão, já que ele é bastante utilizado sobretudo em aplicações relacionadas à segurança. No entanto, também pode ser utilizado em aplicações não relacionadas à segurança física veicular.

Uma mensagem de BSM pode ser transmitida via *unicast*, para apenas um veículo na rede através de comunicação *vehicle-to-vehicle*, ou via *broadcast*, para transmitir possíveis mensagens de alerta para todos os veículos próximos [43].

A principal desvantagem do *Basic Safety Message* está na sua estrutura focada em aplicações relacionadas à segurança. Sendo assim, em outros tipos de aplicações, sobretudo relacionadas a outras áreas de IoT, o BSM pode não mostrar um desempenho satisfatório quando comparado a outros protocolos de propósito geral, que serão vistos adiante.

3.4.2 *Hypertext Transfer Protocol* - HTTP

Este é provavelmente o protocolo à nível de aplicação mais conhecido e amplamente utilizado. o *Hypertext Transfer Protocol* (HTTP) [1] é um protocolo que tem como objetivo a obtenção de recursos como documentos HTML. O HTTP usa uma estrutura de comunicação do tipo cliente-servidor, em que o cliente, que geralmente é um navegador da *web*, envia solicitações ou requisições (*requests*) e recebe mensagens de resposta do servidor (*responses*).

Apesar de ter sido projetado no início dos anos 1990, o HTTP evoluiu bastante ao longo do tempo, se provando um protocolo extensível. Ele é tipicamente enviado sobre o protocolo de transporte TCP, mas pode também usar quaisquer protocolos de transporte confiáveis.

Sua grande vantagem está na sua extensibilidade, pois ele é utilizado não apenas para buscar documentos comuns de texto, como também vídeos, imagens e também publicar conteúdo em servidores através de formulários HTML. Também destaca-se pela facilidade de uso e grande usabilidade, visto que a maioria dos serviços *web* fazem uso do HTTP [54].

3.4.3 *Constrained Application Protocol* - CoAP

O *Constrained Application Protocol* (CoAP) [3] é um protocolo de comunicação a nível de aplicação amplamente utilizado em soluções na área de Internet das Coisas, pois ele é designado normalmente para uso com nós que possuem restrições. O protocolo é projetado para aplicações *device-to-device* (D2D), tais como a comunicação *vehicle-to-vehicle* das Redes Veiculares.

O CoAP segue um modelo de comunicação semelhante ao HTTP, mas com algumas particularidades como a comunicação REST[1]. A Figura 3.3 retrata estas particularidades da comunicação, onde existe uma entidade de *Proxy* REST para ligar a Internet REST, que é a conexão externa fora do ambiente, com o servidor CoAP. Nas comunicações *device-to-device*, o CoAP faz com que ambas as entidades atuem como cliente e servidor ao mesmo tempo. A maneira de funcionamento dos *requests* e *responses* ocorre de forma similar ao HTTP, no entanto, o CoAP lida com esse intercâmbio de informações de maneira assíncrona ao longo de um transporte orientado a datagramas. Por este motivo, o CoAP faz uso de protocolos de transporte não orientados à conexão, como o UDP [66].

Com relação ao formato da mensagem, uma mensagem CoAP começa com um cabeçalho de tamanho fixo de 4 bytes, com as informações de versão, tipo, comprimento de *token*, um código de resposta e um ID de mensagem. Em seguida, a mensagem possui um valor de token de comprimento variável, que pode ter entre 0 a 8 bytes de comprimento, além de valores de *options* e *payload*, que são opcionais [76].

A principal vantagem do protocolo CoAP está na maneira de lidar com soluções que possuam restrições, sejam relacionadas à conectividade, recursos de *hardware* ou consumo de energia. Por isso, este é na maioria das vezes utilizado em projetos IoT, visto que há bastante restrições em todas as áreas, inclusive em Redes Veiculares.

3.4.4 *Message Queuing Telemetry Transport* - MQTT

O protocolo de comunicação à nível de aplicação *Message Queuing Telemetry Transport*, ou *MQ Telemetry Transport* (MQTT) [5] é um protocolo voltado para dispositivos com restrições, como sensores e dispositivos móveis que são otimizados para redes TCP/IP. Por

[1] https://www.devmedia.com.br/introducao-a-web-services-restful/37387

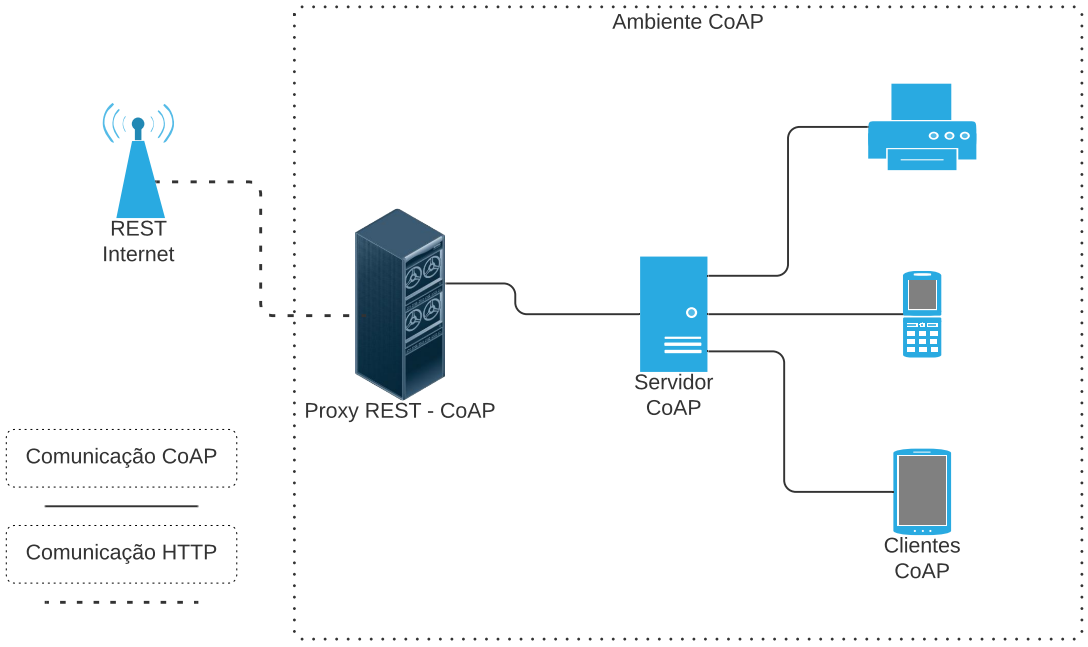

Figura 3.3: Estrutura do protocolo CoAP [34].

ser um protocolo bastante leve e simples, seu esquema de mensagens é baseado no modelo *publish-subscribe*. Com isso, o protocolo visa minimizar o uso da largura de banda, bem como o consumo de recursos dos equipamentos. Sendo assim, garante um bom nível de confiabilidade mesmo com as restrições dos dispositivos.

O protocolo MQTT define dois tipos de entidades na rede: um *message broker* e seus clientes. O *broker* é um servidor que recebe todas as mensagens dos clientes e roteia estas para os clientes de destino relevantes. Um cliente é qualquer entidade que possa interagir com o *broker* e receber mensagens. Um cliente pode ser um objeto IoT ou uma aplicação externa de processamento de dados [20].

Assim como o CoAP, o MQTT é um protocolo amplamente utilizado em comunicações *device-to-device*, sobretudo em aplicações na área de Internet das Coisas. Suas vantagens são as mesmas do protocolo CoAP, no entanto, ele consegue ser ainda mais leve em determinados tipos de aplicações por conta do modelo *publish-subscribe* que usa, diferente do cliente-servidor utilizado no CoAP [67].

Tabela 3.5: Comparação entre protocolos de aplicação.

	BSM	**HTTP**	**CoAP**	**MQTT**
Modelo	Padrão	Cliente-Servidor	Cliente-Servidor	*Publish-Subscribe*
Protocolo de Transporte	WSMP	TCP	UDP	TCP, UDP
Suporte Nativo a Aplicações	Todas	Não Rel. à Segurança	Não Rel. à Segurança	Não Rel. à Segurança

3.4.5 Comparação entre Protocolos de Aplicação

A Tabela 3.5 apresenta um resumo comparativo entre as principais características observadas nos protocolos estudados.

O protocolo BSM, por ter sido criado especificamente para Redes Veiculares, é esperado que apresente bons resultados nos experimentos comparativos. No entanto, espera-se que o protocolo CoAP possa também apresentar resultados positivos, principalmente por fazer uso do UDP como protocolo de transporte. Também se faz útil realizar experimentos com o protocolo MQTT, para demonstrar como o modelo *publish-subscribe* se comportaria no ambiente de Redes Veiculares.

3.4.6 Considerações Finais

Neste capítulo, pôde ser visto os principais simuladores que podem ser utilizados para a simulação de Redes Veiculares, bem como os protocolos de comunicação a nível de aplicação que podem ser avaliados no trabalho.

Por conta da experimentação ser realizada com o foco na camada de aplicação da rede, o componente ideal para ser utilizado é um simulador integrado, tendo em vista que, além da simulação de rede e do tráfego urbano, o mesmo pode trabalhar com estudos em nível de aplicação.

Dentre os simuladores avaliados, destacam-se o *NetSim*, por possuir suporte nativo a protocolos de aplicação, e o *Eclipse Mosaic*, por possuir uma vasta gama de recursos e aplicações nativas que podem ser avaliadas. Por isso, ambos os simuladores integrados foram escolhidos para serem utilizados neste trabalho, e serão detalhados nos capítulos posteriores.

Com relação aos protocolos de aplicação, ganham destaque o BSM, por ser nativo da estrutura de Redes Veiculares, bem como os protocolos CoAP e MQTT, que possuem aplicabilidades em soluções IoT, e por isso são relevantes de serem avaliados neste trabalho. Já

o protocolo HTTP, por ser orientado à conexão, não foi avaliado no experimento, tendo em vista que não possuiria uma aplicabilidade relevante em Redes Veiculares.

Capítulo 4

Avaliação Inicial dos Protocolos IoT em Redes Veiculares

Antes de estudar o desempenho dos protocolos de aplicação IoT em aplicações de Redes Veiculares, faz-se necessário realizar primeiramente uma avaliação inicial destes, com o objetivo de investigar quais possíveis protocolos podem ser posteriormente estudados em aplicações veiculares reais.

Portanto, neste capítulo foi descrita uma avaliação inicial dos protocolos de aplicação em Redes Veiculares. Serão avaliados três protocolos, sendo eles BSM, MQTT e CoAP, com o objetivo de coletar dados preliminares que indiquem a eficácia destes em VANETs, para que os protocolos que tendam a ter resultados satisfatórios possam ser posteriormente avaliados em aplicações veiculares, sejam elas relacionadas ou não à segurança física. No decorrer do capítulo, serão destacadas a metodologia de pesquisa, bem como os cenários de simulação e os resultados comparativos, que são médias das replicações, referentes ao estudo dos protocolos de aplicação.

4.0.1 Ferramentas de Simulação

Enquanto procedimento, esta avaliação foi realizada por meio de ferramentas específicas de simulação na área de pesquisa. Como ferramenta de simulação, foi utilizado o *NetSim Standard* [74], um simulador integrado de rede a eventos discretos direcionado para pesquisas e uso educacional que inclui diversas bibliotecas de tecnologia e permite avaliar protoco-

los e projetos em cenários realistas, otimizar o desempenho de tecnologias e estudar o efeito dos cenários avaliados. Como visto no Capítulo 3, existem outras ferramentas que permitem a simulação de VANETs, tanto simuladores de rede quanto simuladores integrados, como *Veins*, *Ventos* e *NCTUns*, entretanto, a escolha do *NetSim Standard* para esta avaliação preliminar foi realizada por conta da ferramenta possuir suporte nativo a simulações na área de Redes Veiculares, sendo utilizado em diversas pesquisas pela sua completude e adaptabilidade em cenários de VANETs, sobretudo relacionado ao agrupamento de veículos em *platoons*. Também foi preponderante para escolha o fato da ferramenta possuir suporte nativo ao estudo de protocolos de aplicação, característica que outras ferramentas não possuem, facilitando assim a realização de um experimento de caráter inicial, além de ter um excelente suporte da equipe *Tetcos*, desenvolvedora da ferramenta, que sempre esteve à disposição para eventuais dúvidas com o funcionamento do simulador.

Como ferramenta de simulação do tráfego de veículos, foi utilizado o *Sumo - Simulation of Urban Mobility* v1.5.0 [64]. Outras ferramentas, como *MOVE* e *CityMob*, vistas no Capítulo 3, foram consideradas. No entanto, o *Sumo* foi escolhido por ser amplamente utilizado em grande parte das pesquisas na área de Redes Veiculares, facilitando assim sua compreensão e possibilidade de encontrar recursos. O *Sumo* pode funcionar paralelamente em conjunto com o *NetSim*, simulando o mesmo cenário mas tornando possível de gerar um modelo de mobilidade e visualizar seu funcionamento no ponto de vista da mobilidade urbana, enquanto o *NetSim* evidencia a troca de informações entre veículos e outros dispositivos no ponto de vista da topologia de rede.

4.0.2 Padrões de Comunicação Avaliados

O cenário de simulação utilizará como base o padrão de comunicação IEEE 802.11p e a família IEEE 1609, isto é, o trabalho foi feito em cima da pilha de comunicações WAVE [39]. O padrão WAVE já traz consigo um modelo de camadas de rede a ser seguido, e também inclui o padrão DSRC, que prevê comunicações de curto alcance para trocas de informações entre veículos [13]. O WAVE e DSRC também já incluem o seu próprio protocolo de comunicação, o WSMP que é indicado para comunicações *one-hop*, isto é, para transmitir a mensagem a apenas um nó [53].

Neste estudo, foram avaliados os protocolos de aplicação de propósito geral *Constrained*

Application Protocol (CoAP) [23] e *Message Queuing Telemetry Transport* (MQTT) [55]. Todos eles foram testados em uma aplicação de caráter genérico, isto é, que apenas transmite e recebe mensagens sem ter um objetivo específico, embutida no próprio *NetSim*. O protocolo CoAP foi avaliado usando UDP na camada de transporte, dadas as suas características de ser um protocolo não orientado à conexão. Enquanto isso, o protocolo MQTT foi avaliado fazendo uso de TCP na camada de transporte, visto que o mesmo é um protocolo orientado à conexão.

Já com relação à estrutura de comunicação padrão de Redes Veiculares, foi avaliado o protocolo de aplicação BSM [69], fazendo uso do WSMP como protocolo de transporte. Para realizar um estudo comparativo mais eficiente, o BSM foi utilizado na mesma aplicação de caráter genérico em que os outros dois protocolos foram avaliados.

4.0.3 Métricas Avaliadas

Para realizar o estudo comparativo entre os protocolos avaliados, são necessárias métricas que possam compará-los de forma justa.

Neste trabalho, foram usadas três métricas para comparação dos protocolos. São elas:

- *Taxa de transferência (Throughput):* Diz respeito à quantidade de dados transferidos da origem até o destino. Em termos matemáticos, pode-se considerar a taxa de transferência como sendo o número de pacotes enviados dividido pelo tempo que foi gasto. Como unidade, foi utilizado Quilobits por segundo (kbps).

- *Atraso Médio (Delay):* Esta métrica especifica a latência para que os dados viajem pela rede da origem até o destino. Pode-se defini-la, para um conjunto de n pacotes, como o somatório da subtração dos tempos de recebimento pelo tempo de envio de cada pacote, dividido pelo número total de pacotes. Como unidade, foi utilizado Milisegundos (ms).

- *Jitter:* Sendo uma métrica muito conhecida no estudo de redes, é uma variação estatística do atraso na entrega dos dados de uma rede. Em termos matemáticos, pode-se considerar como a variação do atraso entre os pacotes transmitidos. Como unidade, foi utilizado Milisegundos (ms).

4.0.4 Parâmetros da Simulação

Para realizar a comparação dos protocolos de maneira equivalente, é necessário que os parâmetros da simulação sejam iguais em todos os protocolos avaliados. Afinal, caso sejam avaliados com valores de parâmetros diferentes, o estudo comparativo não seria possível de ser realizado.

Neste trabalho, alguns parâmetros equivalentes foram utilizados. São eles:

- *Encriptação (Encryption):* Diz respeito ao tipo de criptografia de dados que o protocolo irá utilizar no processo de comunicação. Como nesta pesquisa não serão estudados aspectos relacionados à segurança dos dados, o parâmetro de encriptação foi marcado como "nenhum".

- *Qualidade do Serviço (Quality of Service, QoS):* Este parâmetro define uma tecnologia responsável por gerir o controle de qualidade da rede. Por meio desta, é possível determinar quais dispositivos e serviços terão maior prioridade de conexão. Neste caso, foi usada o padrão *Best Effort* (BE) como métrica de QoS para todos os protocolos.

- *Distribuição dos pacotes (Distribution of Packets):* O presente parâmetro define a função que irá controlar a maneira de como os pacotes serão distribuídos na rede. Para igualar todos os protocolos, foi usada uma distribuição constante (*Constant*).

- *Distribuição das páginas (Page Distribution):* De maneira semelhante à métrica anterior, esta define uma função que controla o comportamento de como as páginas são distribuídas na rede. Em todos os protocolos, foi usada uma distribuição constante (*Constant*).

- *Tamanho das páginas (Page Size):* Define o tamanho das páginas que serão transmitidas na rede. Em todos os protocolos, foram utilizados um *Page Size* de 40 bytes.

- *Tempo de resposta (Response Time):* Este parâmetro define o tempo mínimo de resposta dos protocolos avaliados. Para serem igualados, o valor foi colocado como 0,001 ms.

- *Resposta Multicast (Multicast Response):* Define se respostas *multicast* estão ou não disponíveis na rede com o determinado protocolo. Em todas as situações, o parâmetro

foi definido com um valor verdadeiro (*True*).

- *Confirmação Ack Requerida (Ack Required):* Este parâmetro define se mensagens *Ack* de confirmação são necessárias na rede. Em todos os protocolos, a mesma foi definida com um valor falso (*False*).

4.1 Cenário de Exemplo

O primeiro cenário avaliado neste trabalho é um exemplo disponível no próprio NetSim. Descrito como *Manhattan Mobility Model*, o modelo possui dez ruas em horizontal e dez em vertical, formando assim um mapa de cidade planejada, com uma topologia em espécie de tabuleiro, que pode ser vista na Figura 4.1.

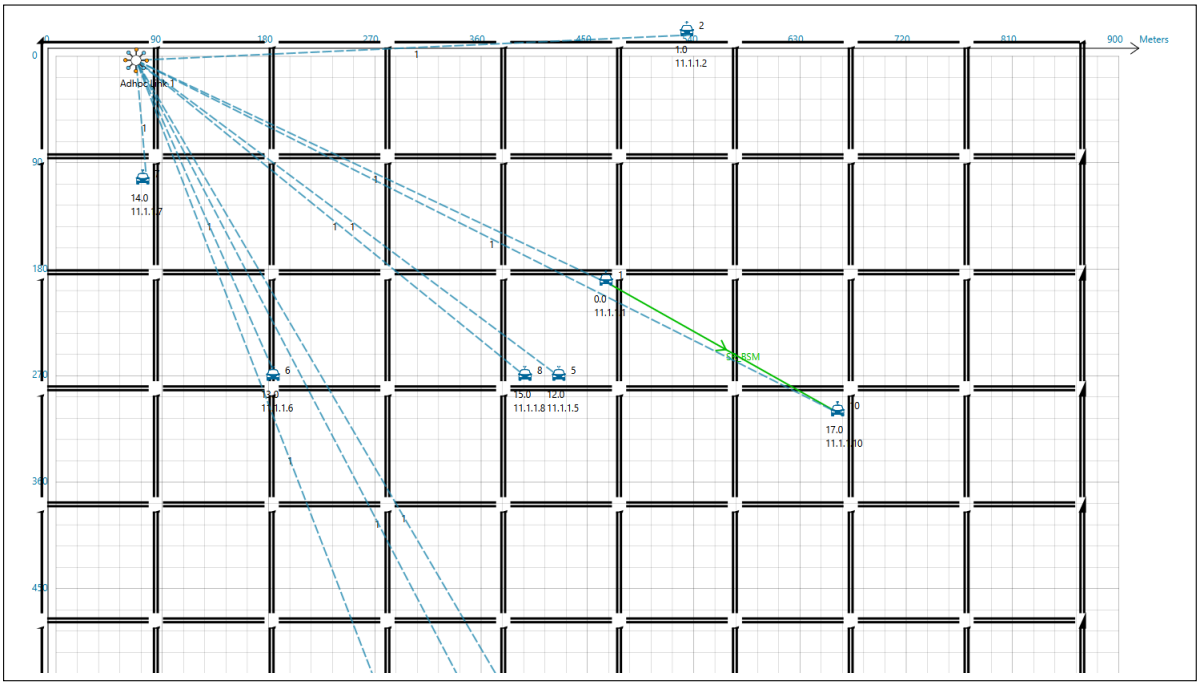

Figura 4.1: Topologia de rede do cenário de exemplo.

Neste modelo, são dispostos dez veículos em lugares aleatórios no mapa, fazendo uso do *Random Way Point* como gerador dos automóveis. Ao todo, a simulação do cenário foi executada 5 vezes, gerando assim 5 replicações. O tempo de duração de cada replicação é de 60 segundos.

Tabela 4.1: Resultados do teste no cenário de exemplo.

	Taxa de transferência (kbps)	Atraso Médio (ms)	Jitter (ms)
BSM	0,080	5,1545	2,0210
MQTT	0,100	20,7900	25,7395
CoAP	0,036	5,7403	5,5004

4.1.1 Resultados

Os resultados do teste são apresentados na Tabela 4.1. Neste primeiro cenário, pode-se perceber que a diferença de desempenho em termos de atraso entre os protocolos BSM e CoAP não foram expressivas, apresentando uma maior diferença em termos de jitter, que houve uma diferença de 3,479 ms, indicando assim que o CoAP tende a oscilar mais que o BSM. Em contraponto, o protocolo MQTT foi mais destoante, apresentando um atraso consideravelmente maior que os outros.

4.2 Cenário de Circuito com RSU

O segundo cenário avaliado neste trabalho envolve o estudo da adição de tecnologias de borda à rede veicular. O cenário de circuito com *Road Site Units* (RSU) simula um total de 5 veículos em um circuito fechado, onde todos eles estão conectados a um RSU na borda da rede. Sendo assim, além das comunicações V2V, este cenário também simula as comunicações *vehicle-to-infrastructure* (V2I). A topologia deste cenário pode ser vista na Figura 4.2.

Assim como no cenário anterior, os veículos fazem uso do *Random Way Point*. Novamente, foram realizadas um total de 5 replicações. O tempo de cada replicação da simulação novamente é de 60 segundos.

4.2.1 Resultados

Os resultados do teste são apresentados na Tabela 4.2. Neste cenário, as conclusões do anterior também se mantêm. O CoAP possui um atraso maior quando comparado ao BSM, embora possua um *jitter* menor, indicando que oscila menos em um circuito fechado cíclico.

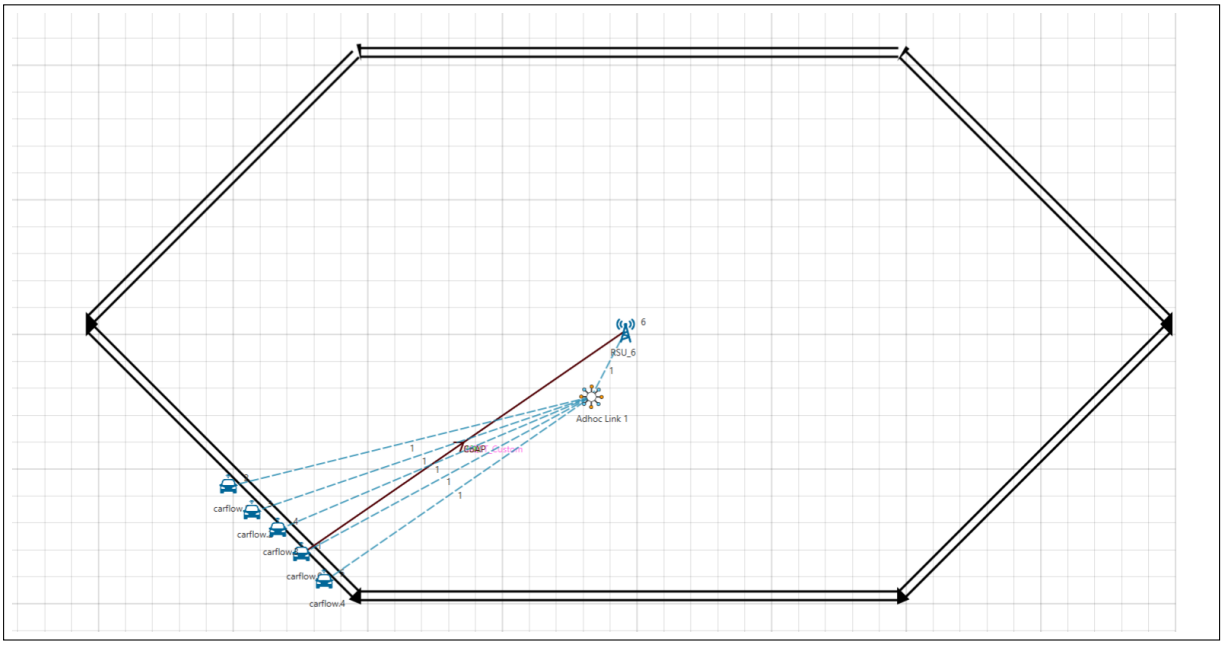

Figura 4.2: Topologia de rede do cenário de circuito com *Road Site Units*.

Tabela 4.2: Resultados do teste no cenário de circuito com RSU.

	Taxa de transferência (kbps)	Atraso Médio (ms)	Jitter (ms)
BSM	0,080	5,1613	2,0330
MQTT	0,100	20,9884	26,0483
CoAP	0,018	11,2308	1,2950

Já o protocolo MQTT novamente não apresenta bons resultados, o que é um indicativo de que ele pode não ser um bom protocolo para ser adotado em aplicações de Rede Veicular.

4.3 Cenário baseado na Cidade de Campina Grande

O terceiro cenário avaliado neste trabalho é mais complexo que o anterior. Foi criado um modelo de mobilidade baseado em Campina Grande, uma cidade do interior da Paraíba. Por ser uma cidade não planejada, diferentemente do cenário anterior em que a organização das ruas eram perfeitas, o cenário acaba se tornando bastante relevante para essa pesquisa. A topologia do cenário pode ser visto na Figura 4.3.

É importante ressaltar que por questões relacionadas a limitações de *hardware*, não foi possível a criação de um modelo de mobilidade da cidade de Campina Grande por inteiro.

Figura 4.3: Modelo de mobilidade de parte da cidade de Campina Grande, visto pelo Sumo.

Sendo assim, um recorte da cidade foi realizado, selecionando os três bairros com maiores fluxos de veículos: Centro, Prata e São José. No modelo, foram dispostos 500 veículos, também aleatoriamente com o uso do *Random Way Point*. Além disso, também foram adicionados três *Road Side Units*, uma em cada bairro. Assim como nos outros cenários, foram realizadas 5 replicações da simulação. O tempo de duração da simulação é de 100 segundos.

4.3.1 Resultados

Os resultados do teste são apresentados na Tabela 4.3. Neste cenário que possui uma maior complexidade no quesito de entidades presentes na rede, pode-se perceber que o BSM possui um atraso bem menor que os demais. O CoAP, apesar de não possuir um atraso tão satisfatório como o BSM, ainda se apresenta na faixa aceitável de atraso em Redes Veiculares [41]. Já a diferença com relação ao *jitter* não é tão significativa entre BSM e CoAP, indicando assim que a oscilação da rede não é tão diferente neste cenário. Em contraponto, o MQTT apresenta um resultado de atraso médio fora da faixa aceitável para VANETs.

Tabela 4.3: Resultados do teste no cenário de Campina Grande.

	Taxa de transferência (kbps)	Atraso Médio (ms)	Jitter (ms)
BSM	0,080	5,2200	2,0090
MQTT	0,100	60,3606	75,1698
CoAP	0,036	15,0293	2,4394

4.4 Cenário com Veículos em *Platoons*

O último cenário a ser avaliado nesta seção visa explorar o conceito de *platoons* visto no Capítulo 2. Neste caso, temos um modelo com apenas 5 veículos. No entanto, eles não foram dispostos de maneira aleatória, pois estão inseridos em um *platoon*. Sendo assim, estes automóveis guardam a mesma distância e se movem na mesma velocidade em um mapa em forma de *loop*, em um mapa semelhante ao segundo cenário, como pode ser visto na Figura 4.4.

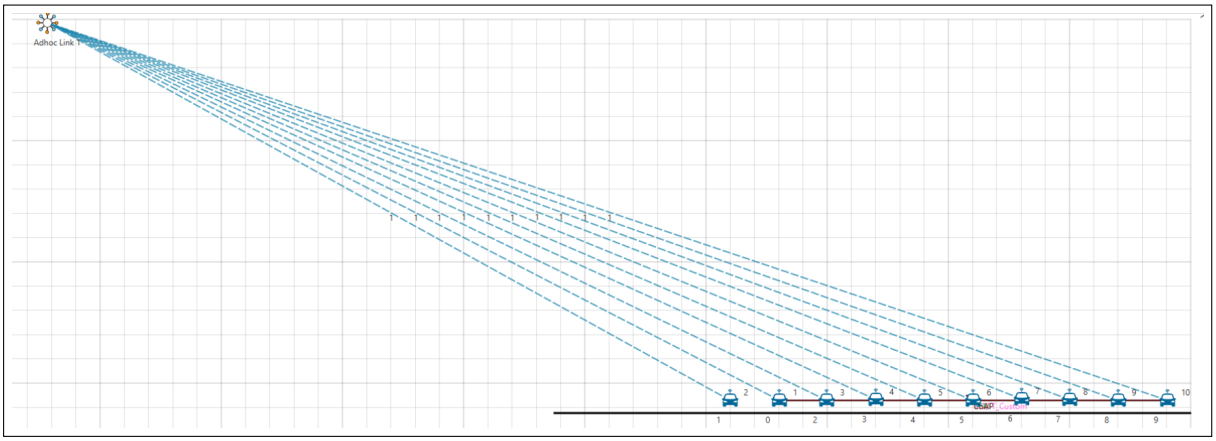

Figura 4.4: Topologia de rede do cenário de um circuito contendo um *platoon* de veículos.

Para simplificar este cenário, em momento algum os veículos deixam o *platoon* avaliado, e também não formam um *platoon* diferente. Eles apenas percorrem o circuito e trocam informações entre si no decorrer dos 60 segundos de simulação. Foram realizadas 5 replicações da simulação.

Tabela 4.4: Resultados do teste no cenário de *platoons*.

	Taxa de transferência (kbps)	Delay (ms)	Jitter (ms)
BSM	0,080	5,1500	1,9670
MQTT	0,100	21,2921	26,4458
CoAP	0,036	8,3340	5,9267

4.4.1 Resultados

Os resultados do teste são apresentados na Tabela 4.4. Pode-se perceber que este último cenário também tem comportamento semelhante aos demais. Novamente, o BSM mostrou um resultado satisfatório com bons valores de atraso e *jitter*, enquanto o CoAP apresentou um atraso maior, mas que ainda se mantém dentro da faixa esperada. Já em questões de *jitter*, percebe-se que o CoAP oscilou consideravelmente mais quando comparado ao BSM. Novamente, o MQTT novamente não apresentou resultados interessantes que justifiquem sua adoção em uma rede veicular. Por fim, o fato dos veículos estarem inseridos em um *platoon* demonstra não influenciar positivamente ou negativamente na avaliação.

4.5 Conclusões

A avaliação preliminar realizada no *NetSim*, embora seja inicial e não conclusiva, serviu para demonstrar possíveis comportamentos dos protocolos quando aplicados em Redes Veiculares.

Primeiramente, como se era esperado, o protocolo BSM apresentou resultados satisfatórios com relação a atraso e *jitter*. Pelo fato de ter sido desenvolvido especificamente para VANETs, é esperado que o mesmo apresente estes resultados. No entanto, o protocolo CoAP também mostrou resultados satisfatórios, principalmente com relação ao atraso médio, onde na maioria dos cenários se manteve abaixo do atraso desejável para VANETs, isto é, abaixo dos 20 ms [41]. Com isso, há um indicativo de resposta para a Questão de Pesquisa 1, tendo em vista que o CoAP mostrou-se viável.

Por outro lado, o protocolo MQTT não apresentou um bom desempenho, o que pode ser explicado por conta de seu modelo *publish-subscribe* com um *broker*, que difere do padrão *cliente-servidor* presente no CoAP. Sendo assim, os resultados preliminares indicam que a

comparação entre protocolos, que é objetivo deste trabalho, se torna viável apenas entre os dois primeiros protocolos citados.

É importante ressaltar que como todas as simulações do *NetSim* são executadas em uma aplicação de caráter genérico, esta limitação não permite maiores conclusões acerca dos resultados. Portanto, o estudo ideal seria avaliar o desempenho destes protocolos tanto em uma aplicação veicular relacionada à segurança física, quanto em uma aplicação não relacionada à segurança, de serviço externo.

Por este motivo, uma nova simulação foi realizada com o intuito de preencher estas lacunas e explorar o desempenho dos protocolos em aplicações veiculares, que será detalhada no capítulo a seguir.

Capítulo 5

Avaliação dos Protocolos BSM e CoAP em Aplicações Veiculares

No capítulo anterior, foi realizada uma avaliação inicial dos protocolos estudados em Redes Veiculares através do *NetSim*. Pelo fato do mesmo possuir suporte nativo a protocolos IoT e uma fácil usabilidade, sua adoção foi importante para a realização do experimento anterior, onde pôde ser concluído que o BSM e o CoAP apresentaram resultados que se enquadram na faixa padrão de Redes Veiculares. No entanto, por conta de suas limitações com relação à ausência de aplicações veiculares, faz-se necessário uma nova experimentação que estude o desempenho dos protocolos que apresentaram bons resultados na avaliação preliminar em aplicações de Redes Veiculares, sejam elas relacionadas ou não à segurança física.

Sendo assim, foi realizada neste capítulo uma avaliação acerca dos protocolos CoAP e BSM avaliados anteriormente com aplicações veiculares relacionadas e não relacionadas à segurança física, com o objetivo de responder às questões de pesquisa deste trabalho.

5.1 Ferramentas de Simulação Utilizadas

Enquanto procedimento, este experimento foi conduzido por meio de ferramentas específicas de simulação na área de pesquisa. Como ferramenta de simulação integrada, foi utilizado o *Eclipse Mosaic*, uma estrutura de simulação multi-domínio e multi-escala para mobilidade conectada e automatizada. Como visto no Capítulo 3, existem outras ferramentas

que permitem a simulação integrada, que integram a simulação de rede como a de tráfego, como *Ventos*, *NCTUns*, *Veins*, e o próprio *NetSim* que foi utilizado no capítulo anterior. Entretanto, a escolha do *Eclipse Mosaic* foi realizada por conta da ferramenta possuir uma estrutura bem dividida em vários módulos de simuladores, incluindo ferramentas de simulação de tráfego, comunicação e aplicações. Além disso, seu módulo de aplicação possibilita avaliar aplicações específicas, sejam elas relacionadas ou não à segurança, vantagem que o mesmo possui quando comparado ao *NetSim*, que por este motivo foi utilizado apenas na avaliação inicial.

Como ferramenta de simulação do tráfego de veículos, foi novamente utilizado o *Sumo*, pelos mesmos motivos explicitados no capítulo anterior. O *Sumo* pode funcionar paralelamente em conjunto com o *Eclipse Mosaic*, assim como ocorria com o *NetSim*.

5.2 Padrões de Comunicação Avaliados

Assim como no capítulo anterior, os cenários avaliados pelo *Eclipse Mosaic* utilizaram como base o padrão de comunicação IEEE 802.11p e a família IEEE 1609, isto é, o trabalho foi feito em cima da pilha de comunicações WAVE, juntamente com o padrão WAVE que já traz consigo um modelo de camadas de rede a ser seguido, incluindo também o padrão DSRC, que provê comunicações de curto alcance para trocas de informações entre veículos. Como protocolo de transporte, novamente foi utilizado o WSMP que é indicado para comunicações *one-hop*, isto é, para transmitir a mensagem a apenas um nó.

Neste estudo, foi avaliado o protocolo de aplicação de propósito geral CoAP, tendo em vista seus bons resultados na simulação anterior. Já com relação à estrutura de comunicação padrão de Redes Veiculares, foi avaliado o protocolo de aplicação BSM, fazendo uso do WSMP como protocolo de transporte. Para realizar um estudo comparativo mais eficiente, ambos foram avaliados em duas aplicações, uma relacionada à segurança e outra não relacionada à segurança, que serão detalhadas posteriormente.

5.3 Métricas Avaliadas

Para realizar o estudo comparativo entre os protocolos avaliados, são necessárias métricas que possam compará-los de forma justa.

Neste trabalho, foram utilizadas quatro métricas para comparação dos protocolos. Duas delas são taxa de transferência (kbps) e atraso médio (ms), que também foram vistas no experimento anterior. Além destas, também foram utilizadas as seguintes métricas:

- *Desvio padrão do atraso médio na amostra:* A métrica diz respeito ao desvio padrão do atraso médio na amostra considerando todas as replicações do cenário. Para padronizar com as outras métricas, foi adotado milisegundos (ms) como unidade de medida.

- *Intervalo de confiança do atraso média na amostra:* Especifica a estimativa do intervalo de confiança (90%) do atraso médio dos protocolos comparados na amostra de replicações realizadas em acada cenário. Por questões de padronização, também foi adotado milisegundos (ms) como unidade de medida.

5.4 Parâmetros da Simulação

Para realizar a comparação dos protocolos de forma equivalente, é necessário que os parâmetros da simulação sejam iguais em todos os protocolos avaliados.

Neste trabalho, os seguintes parâmetros equivalentes foram utilizados:

- *Duração (Duration):* Diz respeito ao tempo de simulação de cada cenário. Todos eles foram colocados com uma duração de 180 segundos, que é o valor padrão do *Mosaic*.

- *Encriptação (Encryption):* Esta métrica define se a comunicação terá algum tipo de criptografia. Como a pesquisa não visa verificar questões de segurança dos dados, a métrica foi definida com um valor falso (*False*).

- *Protocolo de transporte (Transport Protocol):* A presente métrica define o protocolo de transporte que será usada na pilha de comunicação da simulação. Todos os cenários usaram o protocolo UDP.

- *Distribuição das páginas (RandomSeed):* Randomiza o comportamento dos veículos. Em todos os cenários, a variável foi atribuída como 268965854, que é o valor padrão da ferramenta.

- *Tamanho das páginas (Federates):* Diz respeito aos módulos federados que o Eclipse Mosaic irá usar. Os *federates* utilizados, isto é, os que foram definidos como verdadeiro (*True*) foram: *application, environment, omnetpp, output* e *sumo*.

- *Total de replicações:* Esta métrica define o número de replicações que serão realizadas nas comunicações, para ter bons valores de confiança estatística. Em todos os cenários, foram realizadas um total de 1020 replicações.

5.5 Implementação

Um dos aspectos negativos do *Eclipse Mosaic* para a avaliação é o fato do simulador não ter suporte nativo a protocolos de aplicação. Por isso, é necessário uma adaptação em sua estrutura para que seja possível a comparação entre protocolos nos cenários de avaliação.

Para realizar esta adaptação, foi feito uso do módulo *Omnet++ Federate* presente na estrutura de comunicação do *Mosaic*, tendo em vista que este módulo acopla os simuladores Omnet++ e o *framework Inet*, muito úteis para realizar medições que requerem implementações precisas de comunicação.

No entanto, como a estrutura do *Eclipse Mosaic*, bem como o módulo *Omnet++ Federate*, não possuem suporte nativo a protocolos de aplicação, foram feitas adaptações no código para que fosse possível a simulação de uma comunicação CoAP na ferramenta.

Primeiramente, a estrutura do pacote utilizado na comunicação BSM padrão do *Omnet++ Federate* precisou ser avaliada para conversão. O pacote utilizado pelo BSM é genérico e está presente no diretório original do *Inet*. Seu conteúdo é dividido em *Front Popped*, que guarda as informações gerais do pacote, *Data Popped*, que é responsável por armazenar o conteúdo, e *Back Popped*, que possui informações adicionais da comunicação.

Para a construção do pacote CoAP, houve a conversão da estrutura original para um *CoAP Packet*, detalhada na Figura 5.1, em que todas as informações do *Front Popped* foram divididas entre *Version, Type, Token Length, Code* e *Message ID* (linhas 4 à 8 do Código Fonte

5.1), isto é, os campos do cabeçalho CoAP. Já os dados do *Data Popped* se transformaram no *Content CoAP* (linha 9), e o *Back Popped* foi redirecionado para o *Payload* (linha 10), um dos campos opcionais do CoAP. Por fim, o campo *Options* do CoAP, que é opcional, foi deixado em branco. O diagrama de classes do *CoAP Packet* pode ser visto na Figura 5.2.

Código Fonte 5.1: Trecho de código do construtor do *CoAP Packet*

```
CoAPPacket::CoAPPacket(const char *name, const Ptr<const Chunk>& content)
    :
    cPacket(name),
    version(content->getVersion()),
    type(content->getType()),
    tokenLength(content->getTokenLength()),
    code(content->getCode()),
    messageID(content->getMessageID()),
    content(content),
    payload(content->getPayload()),
    totalLength(content->getChunkLength())
{
    constPtrCast<Chunk>(content)->markImmutable();
}
```

Em seguida, foi realizado um mapeamento da classe *MosaicProxyApp*, mostrada no Código Fonte 5.2 e esquematizada na Figura 5.3, onde reside o principal código da comunicação padrão BSM no *Omnet++ Federate*. Neste mapeamento, foi visto que o principal método para envio de pacotes, o *sendPacket*, fazia a criação de um pacote genérico e enviava o mesmo para o método *sendDelayedToUDP*, que criava um pacote UDP e gerava o atraso do envio da comunicação em segundos.

Sendo assim, foi realizado a implementação do mapeamento dessa estrutura para o CoAP, detalhada na Figura 5.4. Após o mapeamento, a classe referida foi adaptada para se comportar como uma comunicação de característica *request-response* de pacotes CoAP. O método *sendPacket* se manteve com a criação do pacote genérico, no entanto, após gerar o atraso do CoAP através do *procDelay*, o método envia esse pacote e o atraso para o novo método *sendDelayedToCoAP*, que instancia o *CoAP Packet* criado contendo a mesma mensagem do pacote BSM, e logo após salva esse pacote no endereço de destino.

Por fim, diferentemente do BSM, o CoAP requer uma mensagem ACK de confirmação.

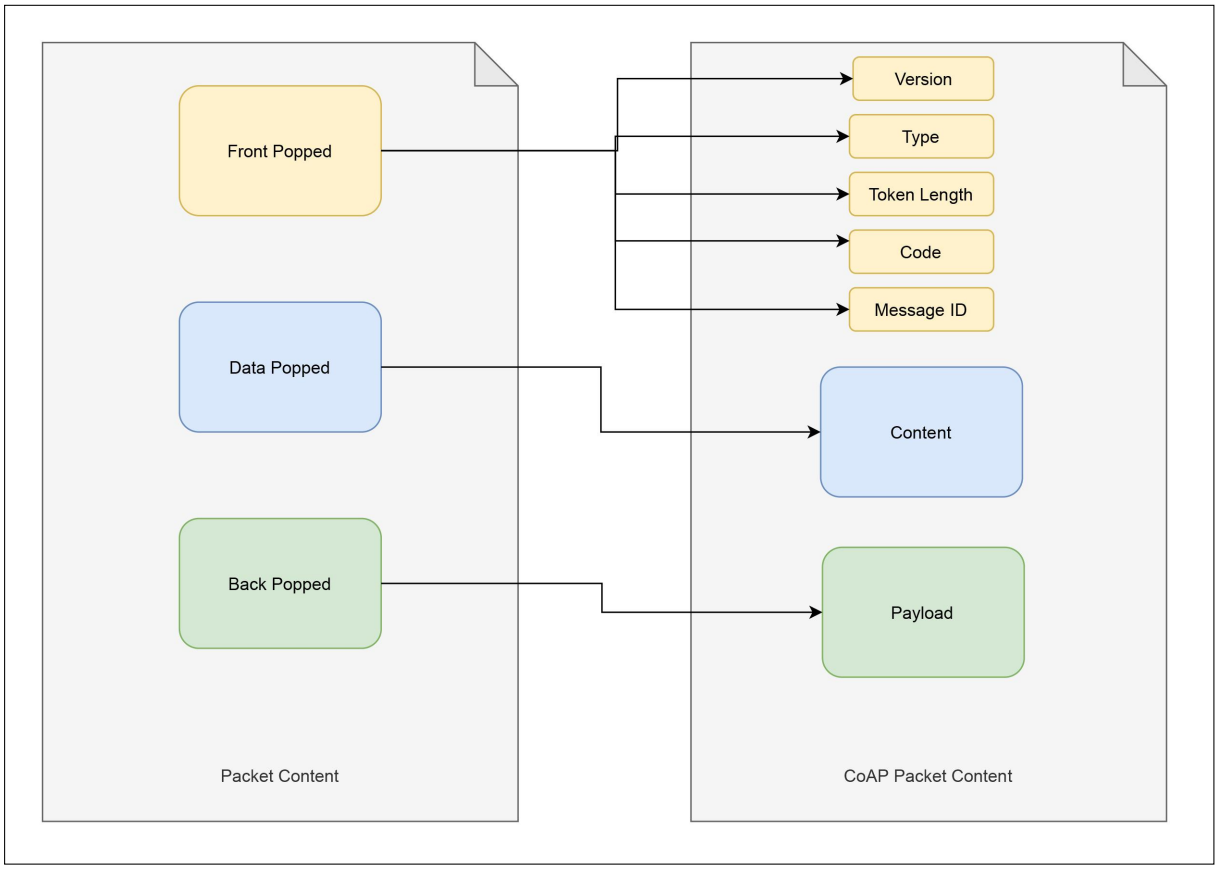

Figura 5.1: Diagrama de conversão para o *CoAP Packet*.

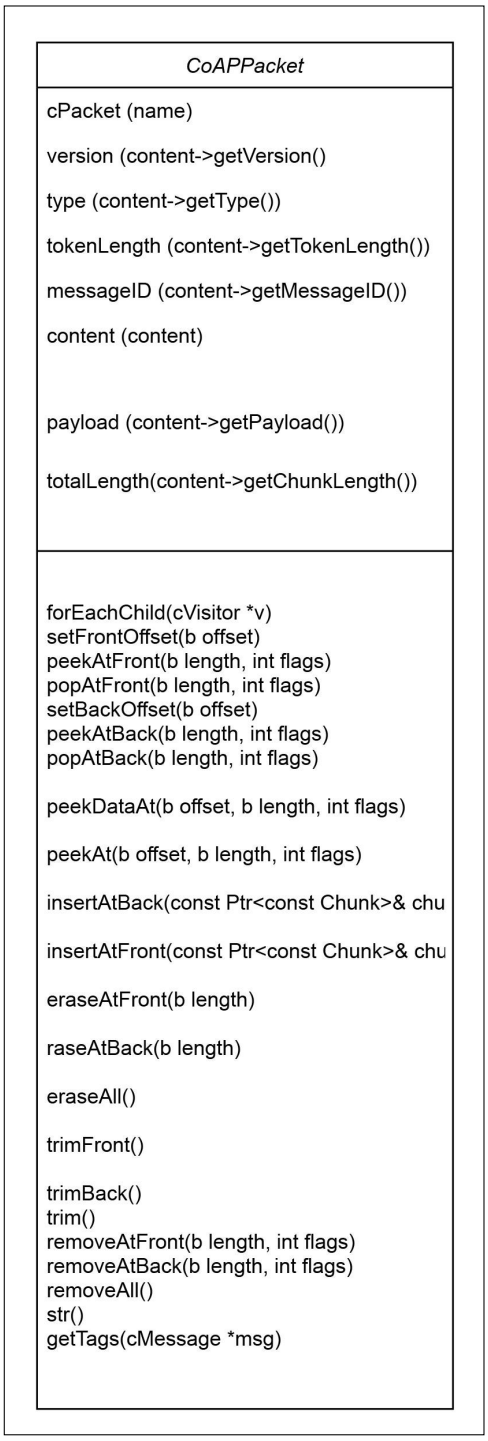

Figura 5.2: Diagrama de classe para o *CoAP Packet*.

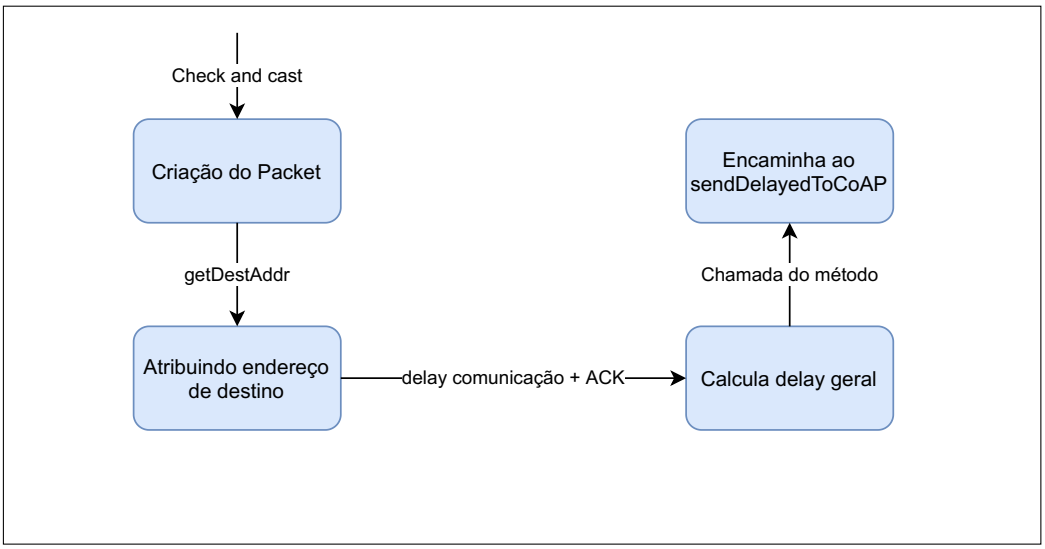

Figura 5.3: Diagrama de fluxo do *sendPacket*.

Para essa implementação, houve uma adaptação no método *receivePacket*, em que além de gerar o pacote de recepção a partir da mensagem, também faz o envio do ACK de recebimento de volta à origem.

Código Fonte 5.2: Trecho de código do método que envia pacotes no *MosaicProxyApp*

```
auto *packet = inet::check_and_cast<MosaicAppPacket *>(msg);
auto destAddress = packet->getDestAddr();

double delay = dbl() * maxProcDelay + delayAck;

sendDelayedToCoAP(omnetpp::check_and_cast<omnetpp::cPacket *>(msg->
    dup()), localPort, destAddress, destPort, delay);
```

5.6 Aplicações Avaliadas

Para realizar o experimento, duas aplicações do *Eclipse Mosaic* foram escolhidas, sendo uma delas relacionada à segurança, e outra não relacionada à segurança. São elas:

- *Comunicação das Luzes do Semáforo (Traffic Light Communication):* Essa aplicação tem seu funcionamento com semáforos de trânsito. Através dela, os *Road Side Units*

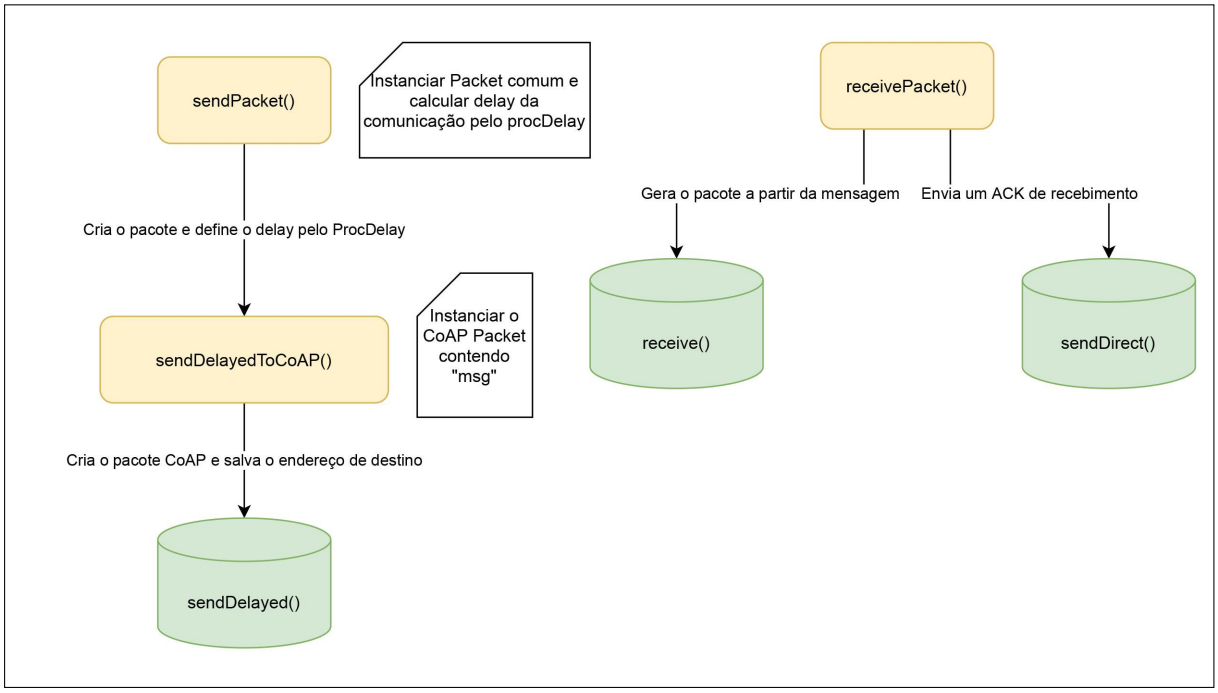

Figura 5.4: Detalhes da implementação da comunicação CoAP.

(RSU) enviam aos veículos próximos uma mensagem informando a cor do semáforo a seguir, podendo ser verde (siga em frente), amarelo (atenção), ou vermelho (necessidade de parar). Os veículos também podem repassar essa mensagem para veículos próximos via *broadcast*. Esta é uma aplicação relacionada à segurança, tendo em vista que impacta diretamente na segurança do motorista e pode evitar acidentes.

- *Informações de Clima (Weather Warning):* A aplicação referida informa aos veículos, através dos *Road Side Units* (RSU), informações de como está o clima na região a seguir, seja ele de chuva, ensolarado ou nublado, além da temperatura. Assim como a primeira aplicação, os veículos também podem repassar as mensagens via *broadcast*. Esta é uma aplicação não relacionada à segurança, já que possui informações para prover mais conforto ao dirigir.

5.7 Cenário baseado na Cidade de Campina Grande

O primeiro cenário avaliado neste experimento é baseado no cenário da cidade de Campina Grande, utilizado na experimentação do *NetSim*. Da mesma forma, foi selecionada uma

área que comporta três bairros da cidade, e dispostos veículos aleatoriamente nas vias.

No entanto, uma alteração foi realizada com o objetivo de deixar o estudo mais realista: a disposição de três *Road Side Units*, cada um posicionado em um bairro da cidade. Em tese, estes *Road Side Units* devem favorecer aplicações não relacionadas à segurança, especialmente as de *infontaiment*, tendo em vista que os RSUs estão posicionados na borda da rede.

Como parâmetros, todos os experimentos foram realizados com duração de 180 segundos, sem criptografia, fazendo uso de UDP como protocolo de transporte, e com os módulos federados de *application, environment, omnetpp, output* e *sumo* ativados no *Mosaic*. Foram realizadas um total de 1020 replicações para cada subcenário, valor estabelecido a partir de um teste A/B de significância estatística.

5.7.1 Cenário com a Aplicação *Traffic Light Communication*

Primeiramente, o cenário foi avaliado com a aplicação *Traffic Light Communication*, que é categorizada como relacionada à segurança, fazendo uso tanto da comunicação padrão do simulador (com protocolo BSM) quanto com o CoAP. Os resultados podem ser vistos na Tabela 5.1.

Os intervalos de confiança do atraso médio dos protocolos comparados neste cenário podem ser vistos graficamente na Figura 5.5. No gráfico, pode-se avaliar que o CoAP possui um atraso consideravelmente maior que o BSM, além de possuir também um intervalo maior de variação com relação às replicações. Além disso, ambos os protocolos se mantêm na faixa ideal para comunicação veicular, o que possibilitaria a adoção do CoAP em um cenário real.

Entretanto, pelo fato da *Traffic Light Communication* se tratar de uma aplicação relacionada à segurança, torna-se mais viável a adoção do BSM, tendo em vista que aplicações deste tipo precisam do mínimo atraso possível, já que muitas vezes estas aplicações podem salvar o motorista do veículo de acidentes, então quanto menor o atraso, mais tempo o mesmo terá para tomar uma possível decisão no trânsito [12].

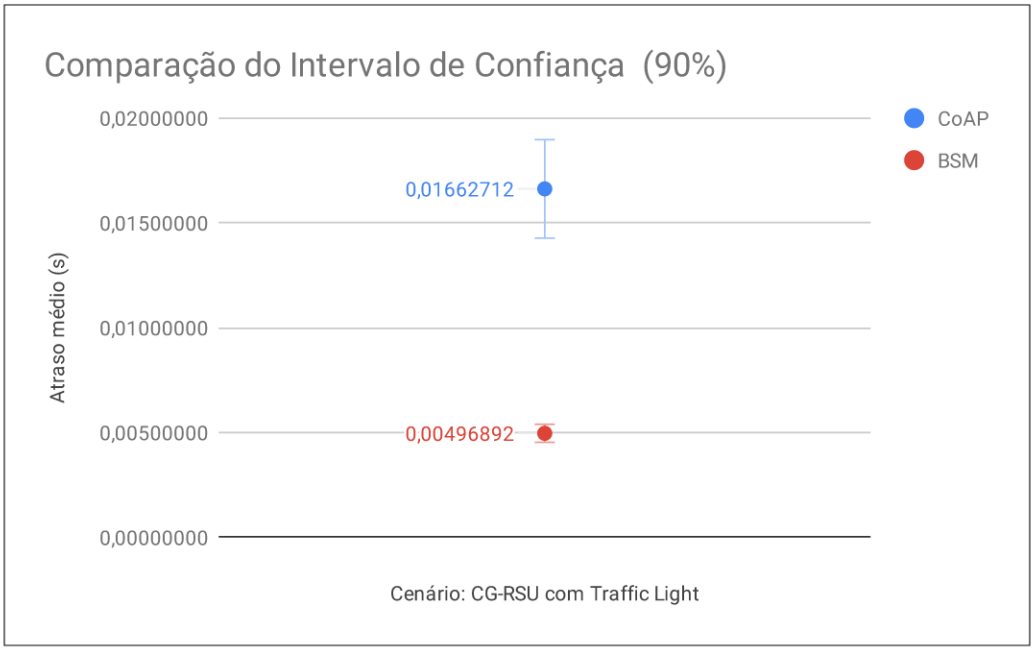

Figura 5.5: Intervalo de confiança do atraso médio do CoAP e BSM no cenário Campina Grande - *Traffic Light Communication*.

Tabela 5.1: Resultados do cenário Campina Grande - *Traffic Light Communication*.

	Atraso médio (ms)	Taxa de transferência média (kbps)	Desvio padrão da amostra (ms)	Intervalo de Confiança (90%) (ms)
CoAP	16,6271	0,249407	15,6375	[14,2791 ; 18,9751]
BSM	4,9689	0,074534	2,8624	[4,5391 ; 5,3987]

5.7.2 Cenário com a Aplicação *Weather Warning*

Além do subcenário anterior, o cenário também foi avaliado com a aplicação *Weather Warning*, que, como foi visto anteriormente, é categorizada como não relacionada à segurança. A experimentação faz uso tanto da comunicação padrão do simulador (com protocolo BSM) quanto com o CoAP. Os resultados podem ser vistos na Tabela 5.2.

Os intervalos de confiança do atraso médio dos protocolos comparados neste cenário podem ser vistos graficamente na Figura 5.6. Neste caso, pode-se verificar que o atraso médio do protocolo CoAP se manteve maior que o do BSM, embora sua variação no intervalo de confiança tenha diminuído quando comparado ao cenário anterior. Isso é um indicativo de que o CoAP se mantém mais estável neste cenário.

Pelo fato da aplicação *Weather Warning* ser não relacionada à segurança, a adoção do CoAP mostra-se como uma escolha viável, por possuir um atraso médio e variações dentro da faixa ideal em comunicações veiculares, além de poder trazer vantagens com relação à interoperabilidade de serviços externos [15].

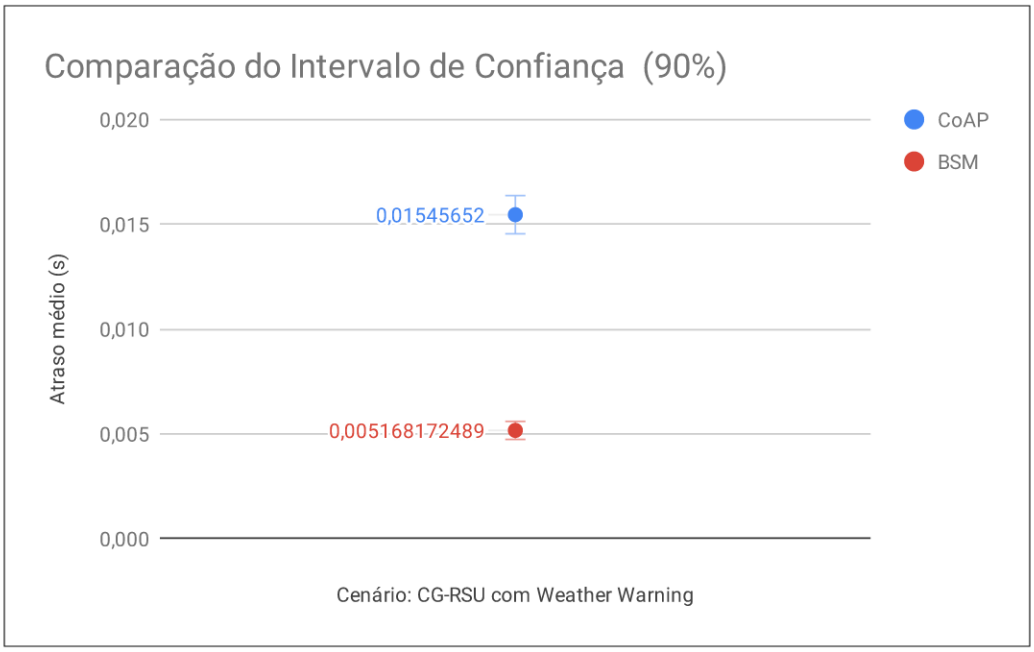

Figura 5.6: Intervalo de confiança do atraso médio do CoAP e BSM no cenário Campina Grande - *Weather Warning*.

Tabela 5.2: Resultados do cenário Campina Grande - *Weather Warning*.

	Atraso médio (ms)	Taxa de transferência média (kbps)	Desvio padrão da amostra (ms)	Intervalo de Confiança (90%) (ms)
CoAP	15,4565	0,231848	6,0697	[14,5451 ; 16,3679]
BSM	5,1682	0,077523	2,9771	[4,7212 ; 5,6152]

5.8 Cenário baseado em Circuito de *Platoons*

O segundo cenário avaliado neste experimento é baseado em um circuito cíclico de cinco veículos em agrupados em um *platoon*, semelhante ao utilizado na experimentação do Net-Sim. Da mesma forma, foi configurado um circuito onde os veículos se movimentam em um *platoon* indefinidamente.

Além disso, um *Road Side Unit* foi posicionado no centro do circuito, com o objetivo de favorecer aplicações não relacionadas à segurança, deixando o RSU posicionado na borda da rede.

Como parâmetros, todos os experimentos foram realizados com duração de 180 segundos, sem criptografia, fazendo uso de UDP como protocolo de transporte, e com os módulos federados de *application, environment, omnetpp, output* e *sumo* ativados no Mosaic. Foram realizadas um total de 1020 replicações para cada subcenário.

5.8.1 Cenário com a Aplicação *Traffic Light Communication*

Assim como anteriormente, este cenário foi avaliado com a aplicação *Traffic Light Communication*, que é categorizada como relacionada à segurança, fazendo uso tanto da comunicação padrão do simulador (com protocolo BSM) quanto com o CoAP. Os resultados podem ser vistos na Tabela 5.3.

Os intervalos de confiança do atraso médio dos protocolos comparados neste cenário podem ser vistos graficamente na Figura 5.7. Pode-se analisar que, neste cenário, a diferença do atraso médio entre o CoAP e o BSM é consideravelmente menor, o que indica que em cenários de menor complexidade e com menor quantidade de veículos conectados na rede, a diferença do atraso entre CoAP e BSM tende a diminuir.

Não obstante, pelo fato da aplicação referida ser relacionada à segurança, a adoção do BSM torna-se a mais viável, mesmo que o CoAP também consiga ser aplicado, tendo em vista que ambos obtiveram resultados dentro da faixa ideal de atraso para Redes Veiculares.

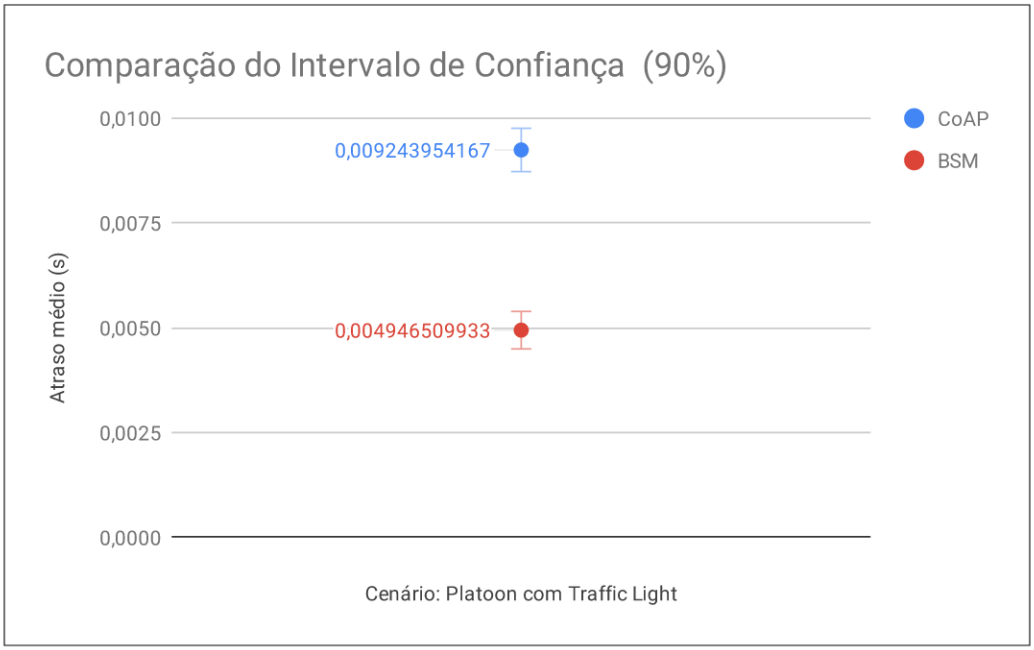

Figura 5.7: Intervalo de confiança do atraso médio do CoAP e BSM no cenário *Platoons - Traffic Light Communication.*

Tabela 5.3: Resultados do cenário *Platoons - Traffic Light Communication.*

	Atraso médio (ms)	Taxa de transferência média (kbps)	Desvio padrão da amostra (ms)	Intervalo de Confiança (90%) (ms)
CoAP	9,2439	0,138659	3,4391	[8,7275 ; 9,7603]
BSM	4,9465	0,074198	2,8299	[4,5216 ; 5,3714]

5.8.2 Cenário com a Aplicação *Weather Warning*

Novamente, o cenário também foi avaliado com a aplicação *Weather Warning*, que, como foi visto anteriormente, é categorizada como não relacionada à segurança. A experimentação faz uso tanto da comunicação padrão do simulador (com protocolo BSM) quanto com o CoAP. Os resultados podem ser vistos na Tabela 5.4.

Os intervalos de confiança do atraso médio dos protocolos comparados neste cenário podem ser vistos graficamente na Figura 5.8. Neste último cenário, pode-se verificar que ambos os protocolos novamente se mantiveram na faixa ideal de atraso para Redes Veiculares. O CoAP obteve um atraso médio um pouco maior quando comparado ao cenário anterior, o que pode ser explicado pelo fato da aplicação *Weather Warning* ser não relacionada à segurança [83].

Por fim, neste cenário, o protocolo CoAP se torna mais viável para adoção, tendo em vista que se a aplicação pode se beneficiar com relação à interoperabilidade, assim como no cenário Campina Grande - *Traffic Light Communication*.

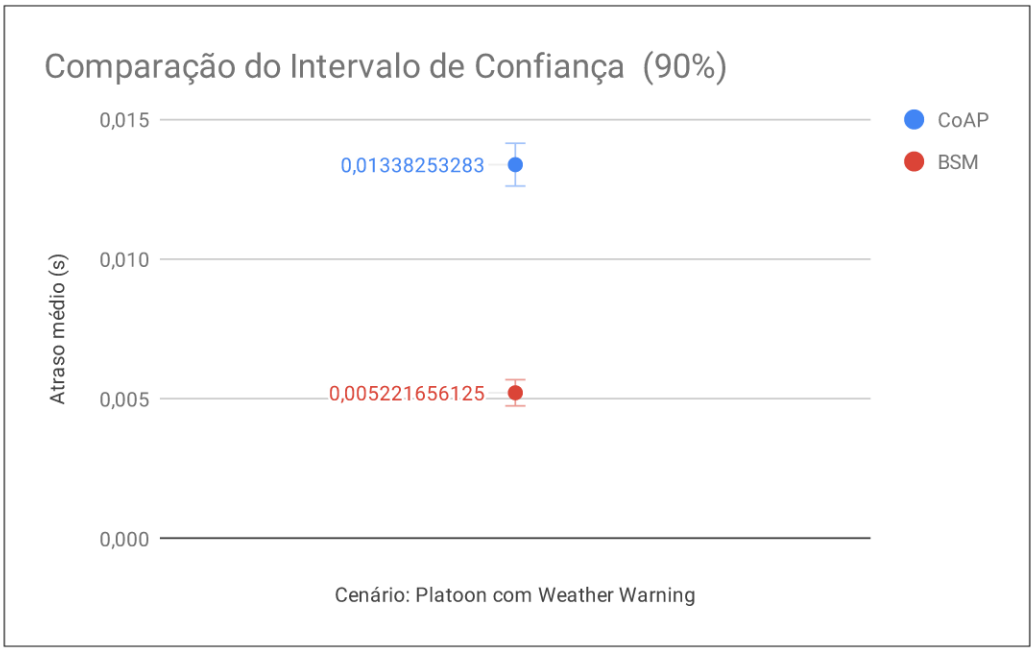

Figura 5.8: Intervalo de confiança do atraso médio do CoAP e BSM no cenário *Platoons - Weather Warning*.

Tabela 5.4: Resultados do cenário *Platoons - Weather Warning*.

	Atraso médio (ms)	Taxa de transferência média (kbps)	Desvio padrão da amostra (ms)	Intervalo de Confiança (90%) (ms)
CoAP	13,3825	0,200738	5,0921	[12,6179 ; 14,1471]
BSM	5,2217	0,078325	3,1174	[4,7537 ; 5,6897]

5.9 Conclusões

Com base nos experimentos realizados, pode-se verificar que os protocolos BSM e CoAP se comportam de maneira semelhante em todos os cenários.

Pela simplicidade de uma comunicação *peer-to-peer*, diferentemente do padrão *request-response* seguido no CoAP, o protocolo BSM tende a ter um atraso menor em todos os cenários avaliados. No entanto, mesmo que essa diferença exista, ambos os protocolos se enquadram dentro da faixa de atraso esperada em comunicações veiculares, tendo em vista que o nível ótimo de atraso médio em uma Rede Veicular gira em torno de 20 milisegundos, enquanto a faixa aceitável se enquadra em torno de 50 milisegundos, como foi visto no Capítulo 2 [41]. Sendo assim, por mais que o CoAP possua um atraso maior, ele ainda se mantém nestas faixas de segurança, respondendo de fato a Questão de Pesquisa 1, já que o CoAP mostrou-se viável em todos os cenários.

Portanto, é possível concluir que em aplicações relacionadas à segurança, como a *Traffic Light Communication*, por conta da importância de ter um atraso médio o mais baixo possível e o CoAP não influenciar positivamente, o BSM seja a solução mais indicada para adoção neste tipo de aplicação.

Entretanto, em aplicações não relacionadas à segurança, como é o caso da *Weather Warning*, o CoAP pode se tornar uma alternativa viável, visto que adotá-lo traz vantagens quanto à interoperabilidade em aplicações externas à Rede Veicular, além de não ser prejudicial quanto ao atraso médio. Com o alto número de dispositivos esperado nos próximos anos através do *mMTC*[1], a interoperabilidade entre dispositivos será crucial, e o protocolo CoAP possui uma alta interoperabilidade em aplicações integradas ao IoT [15]. Dependendo do dispositivo e do tipo de serviço, dois dispositivos diferentes podem não ser interoperáveis sem que haja um protocolo IoT, e o BSM não seria suficiente para aplicações que usassem dispositivos deste tipo.

[1] https://www.mediatek.com/blog/5g-what-are-embb-urllc-and-mmtc

Por fim, foi visto que o cenário em que há um comboio de veículos em *platoons* não tende a afetar os resultados, visto que ambos os protocolos se mantêm na faixa ótima de atraso. Além disso, as comunicações *vehicle-to-infrastructure* providas pelo *Road Side Unit* tendem a trazer vantagens ao CoAP em aplicações não relacionadas à segurança, tendo em vista que provê acesso à borda, facilitando também a interoperabilidade [38].

Tendo em vista estes resultados, faz-se necessário explorar a interoperabilidade entre o protocolo CoAP e uma aplicação veicular não relacionada à segurança que esteja na borda da rede, para validar estas conclusões na prática e confirmar a eficiência da adoção do CoAP neste cenário. Este experimento de validação será apresentado no próximo capítulo.

Capítulo 6

Validação Experimental do Protocolo CoAP em Redes Veiculares

Após os dois experimentos realizados, tanto a avaliação preliminar de protocolos de aplicação em VANETs, quanto a experimentação utilizando aplicações em simuladores, foi concluído que protocolos de aplicação podem ser aplicados à Redes Veiculares e obter um bom desempenho, sobretudo o CoAP, por ter uma estrutura de comunicação UDP de baixa complexidade.

No entanto, ainda se faz necessário realizar uma validação do CoAP em uma implementação com um caso de uso prático, fazendo uso de Computação na Borda, que ajude a justificar o uso deste protocolo, principalmente em aplicações não relacionadas à segurança, as quais mais se beneficiam com serviços na borda [80].

Neste capítulo, foi realizado um experimento de validação funcional, com o objetivo de avaliar o protocolo CoAP em uma comunicação veicular que faz uso da API VIS (*V2X Information Service*) [25] em uma aplicação de serviço de *streaming*.

A API VIS é um serviço do *Multi-access Edge Computing* (MEC) [26] que tem como intuito facilitar a interoperabilidade de comunicações V2X entre várias redes e um ambiente multi-acesso através da troca de informações do usuário com o serviço, agindo como intermediador. Sendo assim, seu principal uso é auxiliar na comunicação de dispositivos em uma Rede Veicular com serviços externos na borda da rede [24]. Esta API foi utilizada como meio de comunicação facilitador entre o *Road Side Unit* e a aplicação *V2X Streaming Service*, que será mais detalhada nas seções a seguir.

6.1 Plano de Validação

Para a realização desta validação, foi necessário primeiramente planejar e esquematizar a implementação de como funcionaria o caso de uso avaliado. Sendo assim, foi pensado um plano de validação a ser seguido, que pode ser visto na Figura 6.1.

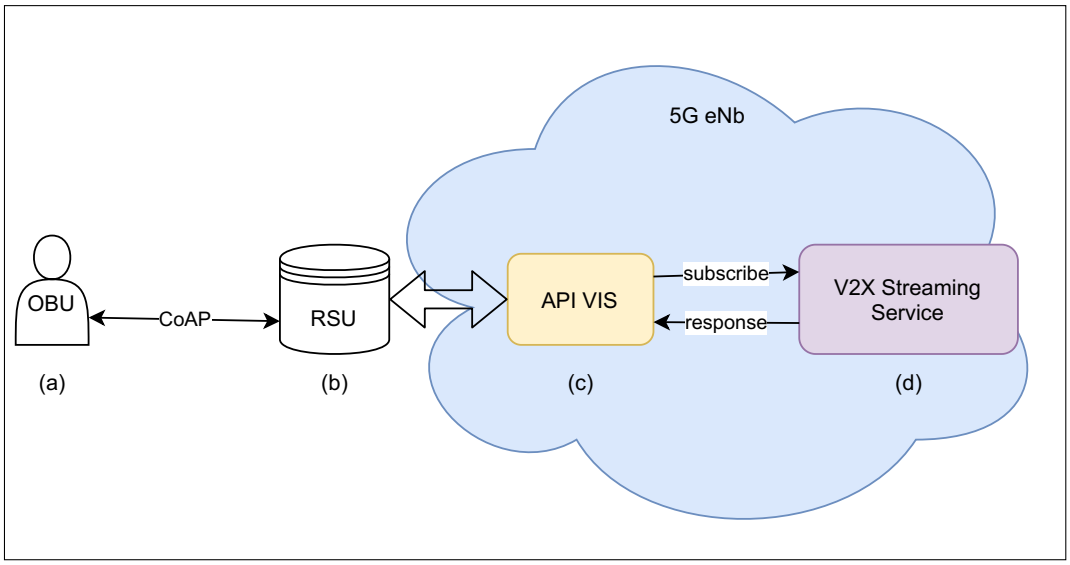

Figura 6.1: Plano de validação do experimento.

O objetivo do experimento é realizar uma comunicação CoAP entre um *On Board Unit* presente em um veículo (Figura 6.1 (a)) e um *Road Side Unit* da infraestrutura de estrada (Figura 6.1 (b)). Este *Road Site Unit*, por sua vez, se comunicaria com a borda da rede através da API *V2X Information Service* (Figura 6.1 (c)), que faria o intermédio das entidades envolvidas.

No plano de validação, a API VIS faz assim a interoperabilidade entre o *Road Site Unit* posicionado na borda da rede e uma aplicação *V2X Streaming Service* (Figura 6.1 (d)), que seria um serviço de *streaming*, categorizado como não relacionado à segurança, favorecendo assim o uso do protocolo CoAP na comunicação. A API, por sua vez, solicita as informações do serviço de *streaming* através de *subscribe*, e recebe um retorno através de *response*.

É importante ressaltar que o *V2X Streaming Service* pode estar em uma rede 5G, que favorece o uso de serviços MEC como o da API VIS [40].

6.2 Requisitos do Experimento

Para nortear o objetivo da validação, os seguintes requisitos foram definidos para o experimento:

- *Requisito 1:* A comunicação do serviço de borda *V2X Streaming Service* com o OBU precisa ocorrer de maneira fim-a-fim, com o protocolo CoAP atuando na comunicação e a API VIS operando como mecanismo de troca de informações entre o ambiente externo e a Rede Veicular.
- *Requisito 2:* A comunicação do OBU com o serviço de borda *V2X Streaming Service* precisa ocorrer de maneira fim-a-fim, com o protocolo CoAP atuando na comunicação e a API VIS operando como mecanismo de troca de informações entre a Rede Veicular e o ambiente externo.

Caso os dois requisitos sejam devidamente atendidos, a validação experimental se dá como concluída. Para cumpri-los, foi necessária uma implementação própria relacionada ao plano de validação, que será detalhada na seção a seguir.

6.3 Implementação

O diagrama apresentado na Figura 6.2 mostra a esquematização de implementação do projeto de validação. A partir da descrição da API VIS, disponível no *Swagger*[1] oferecido pela ETSI, foi realizada a implementação da API a partir da linguagem de programação *Python 3* com o uso do framework *Flask* [60], que auxilia na criação de APIs.

Para este experimento, foi implementado apenas dois serviços da API VIS. O primeiro deles é o *Publish V2X Message*, que é utilizado para a API realizar um *publish* de uma mensagem veicular para outro serviço. Este método é do tipo *Post*, e recebe um arquivo *Json* com os dados de conteúdo da mensagem, formato de codificação, tipo da mensagem e organização padrão [30].

O outro serviço implementado da API VIS foi o *Provisioning Info*, que é utilizado para a API consultar informações de provisionamento para comunicação veicular. O método é do tipo *Get*, e recebe um ID do *On Board Unit* ou *Road Site Unit* a consultar [31].

[1] https://swagger.io/

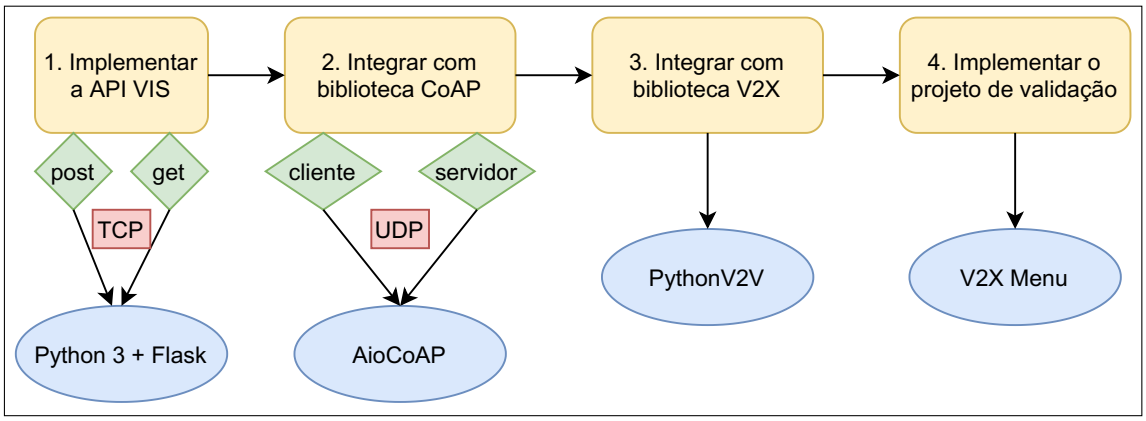

Figura 6.2: Diagrama de implementação do projeto de validação.

Tendo a API VIS implementada, o próximo passo foi integrar a implementação com uma biblioteca de comunicação CoAP. Para isso, foi utilizada a *AioCoAP*[2], por se tratar de uma biblioteca muito completa que faz uso de métodos assíncronos nativos para facilitar as operações simultâneas enquanto mantém uma interface fácil de usar [49]. Além disso, a *AioCoAP* é implementada em *Python 3*, o que facilitou a integração com o projeto de implementação.

A adaptação do CoAP se deu através de duas classes de código: uma representando o cliente CoAP e outra representando o servidor CoAP. Enquanto o cliente CoAP define o caminho de requisições do protocolo para o OBU solicitar informações do serviço na borda, o servidor CoAP define o caminho oposto, isto é, quando o *V2X Streaming Service* envia suas informações em direção ao OBU. Ambas as classes presentes no *AioCoAP* são instanciadas na classe principal do *V2X Menu* e chamadas ao ocorrer comunicações envolvendo as entidades.

Após a integração do código com as classes CoAP da *AioCoAP*, o último passo foi realizar a integração com uma biblioteca de comunicação veicular. A escolhida foi a biblioteca *PythonV2V*[3], pois oferece uma estrutura de fácil entendimento, além de permitir criar veículos e *Road Site Units* para se comunicar [73]. Outra vantagem desta biblioteca é a presença de algumas aplicações, como a aplicação *Streaming Service*, a qual foi utilizada neste experimento. A única desvantagem da *PythonV2V* é sua implementação em *Python 2*, sendo

[2]https://aiocoap.readthedocs.io/en/latest/
[3]https://github.com/islamdiaa/V2V

necessária algumas adaptações para integrar-se ao *Python 3* da implementação.

Para integrar a biblioteca *PythonV2V*, primeiramente foi necessário instanciar as classes *Vehicle* e *RSU*, além de chamar o método que realiza a comunicação entre ambos. Após isso, também foi instanciada na classe principal a aplicação *V2X Streaming Service*, diretamente do módulo *StreamingApp* da biblioteca referida. Por fim, foi realizada a integração entre a comunicação do CoAP e da VIS com as comunicações da biblioteca.

Realizadas todas as integrações, foi implementado o projeto de validação que instancia um *On Board Unit* de um veículo e um *Road Side Unit*, e permite realizar uma comunicação do RSU para o OBU com informações da aplicação *V2X Streaming Service*, e uma comunicação de request do OBU para o RSU, ambas passando pela API VIS. O projeto deste experimento foi chamado de *V2X Menu*.

6.4 Execução do Experimento

Tendo em vista a conclusão da implementação do *V2X Menu*, faz-se necessário executar o experimento. Para obter uma maior fidelidade estatística, cada uma das duas opções de comunicação (serviço de *streaming* para OBU, ou OBU para serviço de *streaming*) foi executada em um total de 150 vezes, com o objetivo de gerar 300 amostras do experimento, uma quantidade suficiente para um nível estatístico aceitável.

Em cada execução do experimento, foram executados na máquina o servidor CoAP da biblioteca *AioCoAP*, bem como a aplicação *Flask* desenvolvida para simular a API VIS. Com estes dois serviços em execução, é possível executar o código principal da implementação do *V2X Menu*.

Na Figura 6.3, é possível visualizar o *V2X Menu* sendo executado em um terminal *Linux*. Ao iniciá-lo, pode-se escolher entre enviar uma mensagem de *streaming* ao veículo, ou requisitar uma informação do veículo para o serviço de *streaming*. Ao escolher uma das opções, são instanciados os objetos do cenário e, em seguida, a comunicação é realizada com sucesso, caso retorne "*Message sent successfully*".

Por fim, o experimento foi executado suscetivas vezes com o objetivo de gerar 300 amostras e ter confiabilidade estatística. Os resultados e conclusões da validação experimental são detalhados na próxima seção.

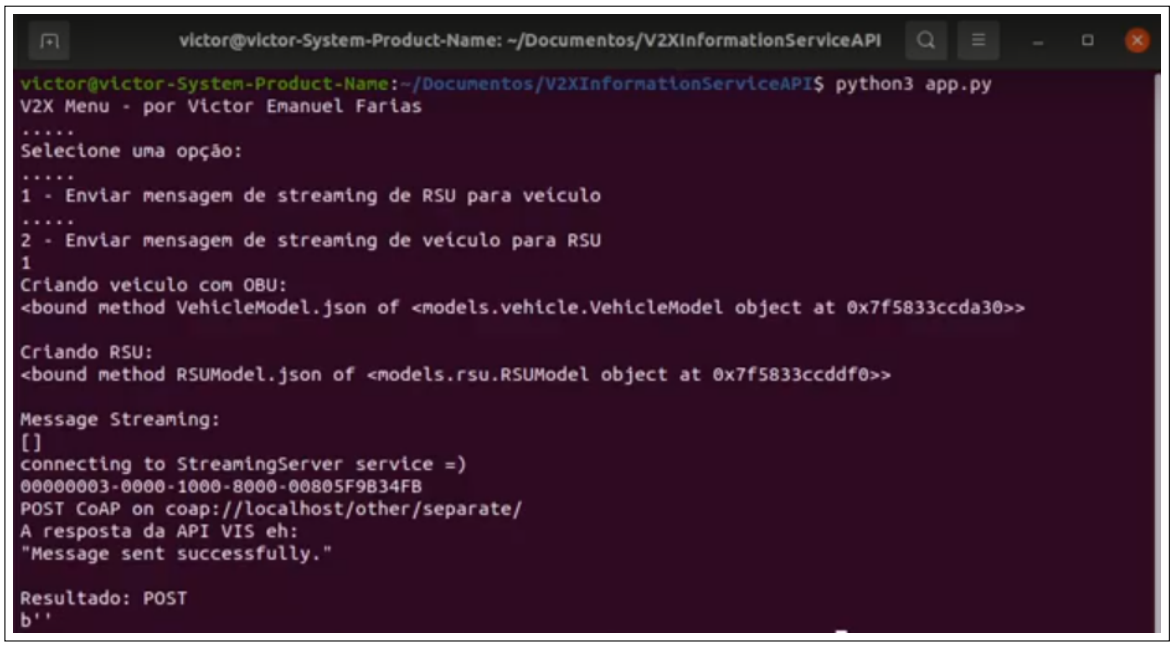

Figura 6.3: Execução do *V2X Menu* no terminal.

6.5 Resultados e Conclusões

A implementação do *V2X Menu* foi executada como prevista nas seções anteriores. Os dois requisitos do experimento foram atendidos, e podem ser vistos de forma detalhada a seguir.

6.5.1 Requisito 1

A comunicação do *V2X Streaming Service* com o OBU foi executada corretamente em todas as amostras da validação experimental. A conexão com o serviço de *streaming*, bem como a comunicação com o CoAP, ocorreram da maneira esperada, onde todas as execuções obtiveram o retorno *"Message sent successfully"* da API VIS. O caminho foi percorrido fim-a-fim seguindo o fluxo do plano de validação, com o protocolo CoAP atuando na comunicação e através do indermédio da API VIS com a função *post* do *Publish V2X Message*.

6.5.2 Requisito 2

A comunicação do OBU com o *V2X Streaming Service* foi executada corretamente em todas as amostras da validação experimental. Assim como no requisito anterior, a conexão

com o serviço de *streaming* e a comunicação com o CoAP ocorreram da maneira esperada, onde todas as execuções também obtiveram o retorno *"Message sent successfully"* da API VIS. O caminho foi percorrido fim-a-fim seguindo o fluxo do plano de validação, com o protocolo CoAP atuando na comunicação e através do indermédio da API VIS com a função *get* do *Provisioning Info*.

6.5.3 Conclusão

Esta validação experimental permite concluir que o CoAP, juntamente com a API VIS, consegue fornecer interoperabilidade entre um serviço externo na borda da rede e os dispositivos veiculares. Portanto, o CoAP é adequado para um cenário de aplicações que se encontram na borda da rede, pois favorece a comunicação com sua grande capacidade de interoperabilidade, a qual é potencializada através do uso de APIs como a *V2X Information Service*. Sendo assim, a validação consegue responder a Questão de Pesquisa 2, pois o CoAP mostra-se adequado em um cenário envolvendo Computação na Borda.

Capítulo 7

Conclusão e Trabalhos Futuros

Este trabalho apresentou uma avaliação de protocolos de aplicação IoT e Computação na Borda em Redes Veiculares, com o objetivo de descobrir se o uso de protocolos IoT em um agrupamento de *platoons* é viável, bem como se eles são vantajosos quando comparados a soluções padrão de VANETs, e se estes são adequados para um cenário de Computação na Borda. Foi apresentada uma visão geral sobre os principais conceitos de Redes Veiculares, bem como foi realizada uma revisão acerca dos protocolos e simuladores que poderiam ser utilizados no experimento.

Na avaliação inicial realizada no simulador *NetSim*, obteve destaque o protocolo CoAP, que mostrou um desempenho semelhante ao padrão de comunicação veicular BSM. O protocolo MQTT, por sua vez, não obteve resultados viáveis. Logo em seguida, o protocolo CoAP e o padrão BSM foram avaliados em aplicações veiculares na ferramenta de simulação *Eclipse Mosaic*. Em ambas as aplicações avaliadas, *Traffic Light Communication* e *Weather Warning*, foi concluído que o protocolo CoAP possui um atraso um pouco maior que o padrão BSM, entretanto ainda se mantém na faixa aceitável de atraso de Redes Veiculares, o que torna viável a sua adoção, principalmente em aplicações não relacionadas à segurança, nas quais ele pode trazer vantagens quanto à interoperabilidade em aplicações externas à Rede Veicular. Sendo assim, a Questão de Pesquisa 1 foi respondida, tendo em vista que as avaliações realizadas comprovam a viabilidade do CoAP quando aplicado em Redes Veiculares.

Para comprovar as vantagens do uso do CoAP quanto à interoperabilidade, foi realizado um experimento de validação funcional fazendo uso da API *V2X Information Service*. Nele,

foi utilizado um serviço de *streaming* em funcionamento na borda da rede, onde a API VIS se comunicava com o *Road Side Unit* que, por sua vez, comunicava-se com o veículo. O experimento respondeu à Questão de Pesquisa 2, comprovando a viabilidade do uso do CoAP em um cenário de Computação em Borda, além de ter mantido um atraso aceitável.

Como trabalho futuro, propõe-se a implementação de um simulador de Redes Veiculares que aborde nativamente a adoção de protocolos de propósito geral e aplicações veiculares e de borda. Tendo em vista que nenhuma das ferramentas de simulação avaliadas possui todas essas funcionalidades, seria de imensa utilidade a criação desta ferramenta proposta no âmbito científico, o que permitiria os mais variados tipos de avaliação.

Bibliografia

[1] Rfc 2616 - hypertext transfer protocol. Disponível em: https://tools.ietf.org/html/rfc2616, 1999. Acesso em: 22/09/2021.

[2] Introducing the internet protocol suite. Disponível em: https://docs.oracle.com/cd/E19455-01/806-0916/6ja85398m/index.html, 2010. Acesso em: 22/09/2021.

[3] Rfc 7252 - the constrained application protocol (coap). Disponível em: https://tools.ietf.org/html/rfc7252, 2014. Acesso em: 22/09/2021.

[4] Vehicle-to-vehicle communications: Readiness of v2v technology for application. White Paper, Agosto 2014.

[5] Mqtt version 5.0. Disponível em: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html, 2019. Acesso em: 22/09/2021.

[6] Omnet++ discrete event simulator. Disponível em: https://omnetpp.org/, 2020. Acesso em: 22/09/2021.

[7] Opnet network simulator. Disponível em: https://opnetprojects.com/opnet-network-simulator/, 2020. Acesso em: 22/09/2021.

[8] Car 2 car communication consortium. Disponível em: https://www.car-2-car.org/, 2021. Acesso em: 22/09/2021.

[9] Iso 21214:2006(en) - intelligent transport systems — communications access for land mobiles (calm). Disponível em: https://www.iso.org/obp/ui/fr/#iso:std:iso:21214:ed-1:v1:en, 2021. Acesso em: 22/09/2021.

[10] Shereen AM Ahmed, Sharifah HS Ariffin, and Norsheila Fisal. Overview of wireless access in vehicular environment (wave) protocols and standards. *environment*, 7:8, 2013.

[11] Mani Amoozadeh, Bryan Ching, Chen-Nee Chuah, Dipak Ghosal, and H Michael Zhang. Ventos: Vehicular network open simulator with hardware-in-the-loop support. *Procedia Computer Science*, 151:61–68, 2019.

[12] V.E.F.C. Borges, D.F.S. Santos, A. Perkusich, and K.M. Malarski. Survey and evaluation of internet of vehicles connectivity challenges. *28th International Conference on Software, Telecommunications and Computer Networks - SoftCOM 2020*, 2020, 2020.

[13] Alves Caio and Freitas Ian. Redes veiculares - dsrc. Disponível em: https://www.gta.ufrj.br/grad/07_2/bernardo/DSRC(2).html. Acesso em: 22/09/2021.

[14] German Aerospace Center. Simulation of urban mobility documentation. https://sumo.dlr.de/docs/index.html, 2020. Acesso em: 22/09/2021.

[15] Nanxing Chen, César Viho, Anthony Baire, Xiaohong Huang, and Jiexi Zha. Ensuring interoperability for the internet of things: Experience with coap protocol testing. *Automatika: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije*, 54(4):448–458, 2013.

[16] Tushar Singh Chouhan and Rajvardhan Somraj Deshmukh. Analysis of dsdv, olsr and aodv routing protocols in vanets scenario: using ns3. In *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, pages 85–89. IEEE, 2015.

[17] Eric Conrad, Seth Misenar, and Joshua Feldman. *CISSP Study Guide*, volume 3. Syngress, 2015.

[18] Juan Contreras-Castillo, Sherali Zeadally, and Juan Antonio Guerrero-Ibañez. Internet of vehicles: architecture, protocols, and security. *IEEE Internet of Things Journal*, 5(5):3701–3709, 2017.

[19] Kusum Dalal, Prachi Chaudhary, and Pawan Dahiya. Performance evaluation of tcp and udp protocols in vanet scenarios using nctuns-6.0 simulation tool. *International Journal of Computer Applications*, 36(6):6–9, 2011.

[20] IBM Developer. Mqtt: Why good for iot. Disponível em: https://developer.ibm.com/br/articles/iot-mqtt-why-good-for-iot/. Acesso em: 30/12/2021.

[21] Kakan Chandra Dey, Anjan Rayamajhi, Mashrur Chowdhury, Parth Bhavsar, and James Martin. Vehicle-to-vehicle (v2v) and vehicle-to-infrastructure (v2i) communication in a heterogeneous wireless network–performance evaluation. *Transportation Research Part C: Emerging Technologies*, 68:168–184, 2016.

[22] Mobilidade Estadão. Mortes no trânsito: Tráfego brasileiro mata 1 pessoa a cada 15 minutos. Disponível em: https://mobilidade.estadao.com.br/mobilidade-com-seguranca/mortes-no-transito-brasileiro-mata-1-pessoa-a-cada-15-minutos/, 2020. Acesso em: 12/10/2021.

[23] Shelby et al. The constrained application protocol (coap). Disponível em: https://datatracker.ietf.org/doc/html/rfc7252. Acesso em: 22/09/2021.

[24] ETSI. Gs mec 030 - v2.1.1 - multi-access edge computing (mec); v2x information service api. Disponível em: https://www.etsi.org/. Acesso em: 22/09/2021.

[25] ETSI. Mec - v2x information service api. Disponível em: https://forge.etsi.org/rep/mec/gs030-vis-api. Acesso em: 22/09/2021.

[26] ETSI. Multi-access edge computing - standards for mec. Disponível em: https://www.etsi.org/technologies/multi-access-edge-computing. Acesso em: 22/09/2021.

[27] Institut Eurecom. Vanetmobisim - newcom. http://vanet.eurecom.fr/, 2007. Acesso em: 22/09/2021.

[28] Elias C Eze, Sijing Zhang, and Enjie Liu. Vehicular ad hoc networks (vanets): Current state, challenges, potentials and way forward. In *2014 20th International Conference on Automation and Computing*, pages 176–181. IEEE, 2014.

[29] Daimler Center for Automotive Information Technology Innovations. Simulation of vehicle-2-x communication - dcaiti - tu berlin. Disponível em: https://www.dcaiti.tu-berlin.de/research/simulation/, 2019. Acesso em: 22/09/2021.

[30] ETSI Forge. Vis - publish_v2x_message. Disponível em: https://forge.etsi.org/swagger/ui/. Acesso em: 22/09/2021.

[31] ETSI Forge. Vis - uu_unicast_provisioning_info. Disponível em: https://forge.etsi.org/swagger/ui/. Acesso em: 22/09/2021.

[32] Eclipse Foundation. Eclipse mosaic - a multi-domain and multi-scale simulation framework for connected and automated mobility. Disponível em: https://www.eclipse.org/mosaic/, 2021. Acesso em: 22/09/2021.

[33] Lorenzo Galati Giordano and Luca Reggiani. *Vehicular Technologies: Deployment and Applications*. BoD–Books on Demand, 2013.

[34] Rinu Gour. 4 major iot protocols — mqtt, coap, amqp, dds. Disponível em: https://medium.com/@rinu.gour123/4-major-iot-protocols-mqtt-coap-amqp-dds-46016897c3e9, 2018. Acesso em: 22/09/2021.

[35] Jose Grimaldo and Ramon Martí. Performance comparison of routing protocols in vanets under black hole attack in panama city. In *2018 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pages 126–132. IEEE, 2018.

[36] Ge Guo and Shixi Wen. Communication scheduling and control of a platoon of vehicles in vanets. *IEEE Transactions on intelligent transportation systems*, 17(6):1551–1563, 2015.

[37] Jérôme Härri, Fethi Filali, Christian Bonnet, and Marco Fiore. Vanetmobisim: generating realistic mobility patterns for vanets. In *Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, pages 96–97, 2006.

[38] Xumin Huang, Rong Yu, Miao Pan, and Lei Shu. Secure roadside unit hotspot against eavesdropping based traffic analysis in edge computing based internet of vehicles. *IEEE Access*, 6:62371–62383, 2018.

[39] IEEE. Ieee standard for wireless access in vehicular environments (wave). Technical Report 1609.12-2019, IEEE Standards Association, Piscatway, USA, 2019.

[40] Teodor B Iliev, Gr Y Mihaylov, Tsviatko D Bikov, Elena P Ivanova, Ivaylo S Stoyanov, and DI Radev. Lte enb traffic analysis and key techniques towards 5g mobile networks. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 497–500. IEEE, 2017.

[41] Wen-Quan Jin and Do-Hyeun Kim. Implementation and experiment of coap protocol based on iot for verification of interoperability. *The journal of the institute of internet, broadcasting and communication*, 14(4):7–12, 2014.

[42] Florent Kaisser, Christophe Gransart, and Marion Berbineau. Simulations of vanet scenarios with opnet and sumo. In *International Workshop on Communication Technologies for Vehicles*, pages 103–112. Springer, 2012.

[43] Seungmo Kim and Byung-Jun Kim. Prioritization of basic safety message in dsrc based on distance to danger. *arXiv preprint arXiv:2003.09724*, 2020.

[44] N. Kleina. Afinal, por que ainda não temos 5g no brasil? Disponível em: https://www.tecmundo.com.br/dispositivos-moveis/139255-ainda-nao-temos- 5g-brasil.htm, 2019. Acesso em: 22/09/2021.

[45] Juliana Kolb. Modelo osi (open systems interconnection). Disponível em: http://jkolb.com.br/modelo-osi-open-systems-interconnection/, 2016. Acesso em: 22/09/2021.

[46] Damigou Kombate et al. The internet of vehicles based on 5g communications. In *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 445–448. IEEE, 2016.

[47] Kun-Chan Lan. Move: A practical simulator for mobility model in vanet. Disponível em: https://www.igi-global.com/chapter/move-practical-simulator-mobility-model/39536, 2010. Acesso em: 22/09/2021.

[48] Wenshuang Liang, Zhuorong Li, Hongyang Zhang, Shenling Wang, and Rongfang Bie. Vehicular ad hoc networks: architectures, research issues, methodologies, challenges, and trends. *International Journal of Distributed Sensor Networks*, 11(8):745303, 2015.

[49] Christian Amsüss Maciej Wasilak. aiocoap – the python coap library. Disponível em: https://aiocoap.readthedocs.io/en/latest/. Acesso em: 22/09/2021.

[50] Rahul Mangharam, Daniel Weller, Raj Rajkumar, Priyantha Mudalige, and Fan Bai. Groovenet: A hybrid simulator for vehicle-to-vehicle networks. In *2006 Third Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services*, pages 1–8. IEEE, 2006.

[51] Francisco J Martinez, J-C Cano, Carlos T Calafate, and Pietro Manzoni. Citymob: a mobility model pattern generator for vanets. In *ICC Workshops-2008 IEEE International Conference on Communications Workshops*, pages 370–374. IEEE, 2008.

[52] Sajjad Akbar Mohammad, Asim Rasheed, and Amir Qayyum. Vanet architectures and protocol stacks: a survey. In *International Workshop on Communication Technologies for Vehicles*, pages 95–105. Springer, 2011.

[53] Yasser L Morgan. Managing dsrc and wave standards operations in a v2v scenario. *International Journal of Vehicular Technology*, 2010, 2010.

[54] Mozilla. Uma visão geral do http. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview, 2020. Acesso em: 22/09/2021.

[55] MQTT.org. Mqtt: The standard for iot messaging. Disponível em: https://mqtt.org/. Acesso em: 22/09/2021.

[56] Robert Nagel and Stephan Eichler. Efficient and realistic mobility and channel modeling for vanet scenarios using omnet++ and inet-framework. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–8, 2008.

[57] NSNam. The network simulator - ns - 2. Disponível em: https://www.isi.edu/nsnam/ns/, 2020. Acesso em: 22/09/2021.

[58] NSNam. Ns-3, a discrete-event network simulator for internet systems. Acesso em: https://www.nsnam.org/, 2020. Acesso em: 22/09/2021.

[59] School of Computer and Communication Sciences EPFL. Trans - traffic and network simulation environment. Disponível em: http://trans.epfl.ch. Acesso em: 22/09/2021.

[60] Pallets. Flask - web development, one drop at a time. Disponível em: https://flask.palletsprojects.com/en/2.0.x/. Acesso em: 22/09/2021.

[61] Ytallo Pessoa. Avaliação de desempenho de redes veiculares ad hoc(vanets) definidas por software. 2018.

[62] Michal Piorkowski, Maxim Raya, A Lezama Lugo, Panagiotis Papadimitratos, Matthias Grossglauser, and J-P Hubaux. Trans: realistic joint traffic and network simulator for vanets. *ACM SIGMOBILE mobile computing and communications review*, 12(1):31–33, 2008.

[63] Robert Protzmann, Björn Schünemann, and Ilja Radusch. Simulation of convergent networks for intelligent transport systems with vsimrti. *Networking Simulation for Intelligent Transportation Systems: High Mobile Wireless Nodes*, pages 1–28, 2017.

[64] Chitraxi Raj, Urvik Upadhayaya, Twinkle Makwana, and Payal Mahida. Simulation of vanet using ns-3 and sumo. *International Journal of Advanced Research in Computer Science and Software Engineering*, 4(4), 2014.

[65] Divya Rathi and Rashmi Ranade. Performance evaluation of aodv routing protocol in vanet with ns2. *IJIMAI*, 4(3):23–27, 2017.

[66] Digvijaysinh Rathod and Sunit Patil. Security analysis of constrained application protocol (coap): Iot protocol. *International Journal of Advanced Studies in Computers, Science and Engineering*, 6(8):37, 2017.

[67] Margaret Rouse. Mqtt (mq telemetry transport). Disponível em: https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport, 2016. Acesso em: 22/09/2021.

[68] Ghassan Samara, Wafaa AH Al-Salihy, and R Sures. Security issues and challenges of vehicular ad hoc networks (vanet). In *4th International Conference on New Trends in Information Science and Service Science*, pages 393–398. IEEE, 2010.

[69] Suryansh Saxena and Isaac K Isukapati. Simulated basic safety message: Concept & application. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2450–2456. IEEE, 2019.

[70] Keith Shaw. The osi model explained: How to understand (and remember) the 7-layer network model. Disponível em: https://www.networkworld.com/article/3239677/the-osi-model-explained-how-to-understand-and-remember-the-7-layer-network-model.html, 2018. Acesso em: 22/09/2021.

[71] Christoph Sommer. Veins - the open source vehicular network simulation framework. Disponível em: https://veins.car2x.org/, 2020. Acesso em: 22/09/2021.

[72] Ali Syed, George Yin, Abhilash Pandya, Hongwei Zhang, et al. Coordinated vehicle platoon control: Weighted and constrained consensus and communication network topologies. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 4057–4062. IEEE, 2012.

[73] Islam El Tayar. Pythonv2v. Disponível em: https://github.com/islamdiaa/V2V. Acesso em: 22/09/2021.

[74] Tetcos. Netsim - network simulator and emulator. Disponível em: https://www.tetcos.com/index.html, 2020. Acesso em: 22/09/2021.

[75] F. Todd. Connected cars: Ten companies using the internet of things to improve driving experience. Disponível em: https://www.ns-businesshub.com/technology/connected-cars-iot/, 2018. Acesso em: 22/09/2021.

[76] UFRJ. Constrained application protocol - funcionamento do coap. Disponível em: https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/coap/funcionamento.html. Acesso em: 30/12/2021.

[77] Unex. Obu-301u - dsrc us stack. Disponível em: https://www.unex.com.tw/products/v2x/v2vsolution/on-board-unit/1609x-stack-preload/detail/obu-301u, 2020. Acesso em: 22/09/2021.

[78] Davis University of California. Ventos - vehicular network open simulator. Disponível em: http://trans.epfl.ch, 2017. Acesso em: 22/09/2021.

[79] GRC UPV. Citymob: Generador de patrones de movimiento para entornos vanet. Disponível em: http://www.grc.upv.es/Software/oldsw/citymob.html, 2009. Acesso em: 22/09/2021.

[80] Kai Wang, Hao Yin, Wei Quan, and Geyong Min. Enabling collaborative edge computing for software defined vehicular networks. *IEEE Network*, 32(5):112–117, 2018.

[81] Pengfei Wang, Boya Di, Hongliang Zhang, Kaigui Bian, and Lingyang Song. Platoon cooperation in cellular v2x networks for 5g and beyond. *IEEE Transactions on Wireless Communications*, 18(8):3919–3932, 2019.

[82] Shie-Yuan Wang and CL Chou. Nctuns 5.0 network simulator for advanced wireless vehicular network researches. In *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, pages 375–376. IEEE, 2009.

[83] Qing Xu, Tony Mak, Jeff Ko, and Raja Sengupta. Vehicle-to-vehicle safety messaging in dsrc. In *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, pages 19–28, 2004.

[84] Fangchun Yang, Shangguang Wang, Jinglin Li, Zhihan Liu, and Qibo Sun. An overview of internet of vehicles. *China communications*, 11(10):1–15, 2014.

Apêndice A

Trecho de código da implementação da API VIS

```
1     def get(self, credential):
2
3         v2xmsg = V2XMsg.find_message(credential)
4         if v2xmsg:
5             return v2xmsg
6         return {'message': 'Provisioning info not found.'}, 404 # not
              found
7
8     def post(self, credential):
9
10        dados = V2XMsg.argumentos.parse_args()
11
12        v2x_message = {
13        'msgContent': dados['msgContent'],
14        'msgEncodeFormat': dados['msgEncodeFormat'],
15        'msgType': dados['msgType'],
16        'stdOrganization': dados['stdOrganization']
17        }
18
19        if isNulo(v2x_message):
20            return "No Content", 204
21        elif credential not in lista_elementos:
22            erro = {
```

```
23              'detail': 'used when the client did not submit credentials.',
24              'instance': 'Error',
25              'status': 0,
26              'title': 'Unauthorized',
27              'type': 'Error 401'
28              }
29              return erro, 401
30          elif credential == None:
31              erro = {
32              'detail': 'used when a client provided a URI that cannot be
                    mapped to a valid resource URI.',
33              'instance': 'Error',
34              'status': 0,
35              'title': 'Not Found',
36              'type': 'Error 404'
37              }
38              return erro, 401
39          else:
40              lista_v2xmsg.append(v2x_message)
41              return "Message sent successfully.", 200
```

Apêndice B

Trecho de código da implementação do cliente *AioCoAP*

```
1
2  logging.basicConfig(level=logging.INFO)
3
4  async def main():
5
6      protocolo = await Context.create_client_context()
7      requisicao = Message(code=GET, uri='coap://localhost/other/separate')
8
9      try:
10         response = await protocolo.request(requisicao).response
11     except Exception as e:
12         print('Falha ao buscar recurso: ')
13         print(e)
14     else:
15         print('Resultado: %s\n%r'%(response.code, response.payload))
```