
Pós-Graduação em Ciência da Computação

**Maria Atrícia Sabino Maciel**

# SDVN-vC - UMA ARQUITETURA FLEXÍVEL COM SUPORTE À ESCALABILIDADE PARA REDES VEICULARES DEFINIDAS POR SOFTWARE USANDO NFV

Recife

2018

**Maria Atrícia Sabino Maciel**

# SDVN-vC - UMA ARQUITETURA FLEXÍVEL COM SUPORTE À ESCALABILIDADE PARA REDES VEICULARES DEFINIDAS POR SOFTWARE USANDO NFV

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

**Orientador:** Prof. Kelvin Lopes Dias

Recife
2018

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

M152s Maciel, Maria Atrícia Sabino
SDVN-vC - uma arquitetura flexível com suporte à escalabilidade para redes veiculares definidas por software usando NFV / Maria Atrícia Sabino Maciel. – 2018.
74 f.: il., fig., tab.

Orientador: Kelvin Lopes.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2018.
Inclui referências.

1. Redes de computadores. 2. Redes definidas por software. I. Lopes, Kelvin (orientador). II. Título.

004.6 CDD (23. ed.) UFPE- MEI 2018-110

**Maria Atrícia Sabino Maciel**

# SDVN-vC - UMA ARQUITETURA FLEXÍVEL COM SUPORTE À ESCALABILIDADE PARA REDES VEICULARES DEFINIDAS POR SOFTWARE USANDO NFV

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Aprovada em: 14 de Junho de 2018.

**BANCA EXAMINADORA**

---

Prof. Dr. Kelvin Lopes Dias (Orientador)
Universidade Federal de Pernambuco

---

Prof. Dr. Nelson Souto Rosa (Examinador Interno)
Universidade Federal de Pernambuco

---

Prof. Dr. Danielo Gonçalves Gomes (Examinador Externo)
Universidade Federal do Ceará

*Dedico este trabalho aos meus pais que, apesar das dificuldades, sempre fizeram de tudo para que eu e minha irmã estudássemos nas melhores universidades do País.*

# AGRADECIMENTOS

Para alcançarmos um objetivo, lutamos e dedicamos um tempo considerável de nossas vidas. Ao final dos ciclos, depois dos momentos de muito trabalho, dedicação, desgastes e até a privação, posso afirmar que sempre vale a pena. Porém, se cheguei até aqui foi com a ajuda divina e de várias pessoas, afinal, não se pode vencer sozinho na pesquisa e nem na vida, então, nada mais justo do que gradecer a essas pessoas que ajudaram nessa minha conquista.

Em primeiro lugar, agradeço a Deus pela vontade de continuar e superar cada obstáculo. Por me mostrar o bem em meio a tanto mal. Durante todo esse tempo, querido Senhor, percebi quão importante e necessária é sua orientação em minha vida. Nunca me cansarei de lhe agradecer, e graças ao seu exemplo de vida, consegui me tornar uma pessoa melhor.

Agradeço a toda minha família, em especial aos meus pais, Alcilene e Valdeci, a minha irmã Patrícia e meus avós Adalgisa (in memoriam) e Francisco Sabino, que ao longo dessa caminhada sempre estiveram ao meu lado, independente das tribulações. Agradeço pelos ensinamentos que ajudaram na construção do meu caráter e também todos os esforços para me proporcionar uma educação de qualidade.

Agradeço ao meu professor e orientador Kelvin Dias, por acreditar e apostar na minha capacidade e no potencial deste trabalho, proporcionando meios para torná-lo um trabalho científico. Agradeço por toda dedicação, disponibilidade e principalmente a paciência no decorrer de todo o mestrado. Também não poderia deixar de agradecer pelos "puxões de orelha" devidamente dados durante esse período, que me fizeram amadurecer pessoal e profissionalmente.

Agradeço também aos meus amigos pelo apoio emocional, aos amigos do CIn e do grupo de pesquisa que me ajudaram bastante com discussões e contribuições, obrigada por todos os conselhos dados durante essa caminhada.

Agradeço a todos os professores do CIn pelo conhecimento compartilhado e pelo exemplo de dedicação e comprometimento. Aos servidores do CIn por todo o suporte necessário para a realização desta pesquisa, em especial ao amigo Adriano Henrique.

# RESUMO

Com o advento das redes veiculares (Vehicular Ad Hoc Networks - VANETs), serviços inovadores podem ser viabilizados para proporcionar maior conforto aos passageiros, prevenção de acidentes e melhorar a mobilidade em grandes centros urbanos. Na comunicação VANET denominada veículo à infraestrutura (vehicle-to-infrastructure - V2I), o veículo troca informações com pontos de acesso sem fio dispostos ao longo da via. Com a arquitetura atual, alguns desafios em VANET requerem mais pesquisas e novas soluções em função do aumento significativo de dispositivos conectados, necessidade de uso eficiente dos recursos, oscilação no tráfego de informações, atrasos e perdas de pacotes devido à alta mobilidade dos veículos, conexões não confiáveis, e requisitos de qualidade de serviço (Quality of Service - QoS) das aplicações. Para lidar com os desafios do gerenciamento de VANETs, a literatura da área tem intensificado o emprego de redes definidas por software (Software Defined Networking – SDN), que baseiam-se na separação entre os planos de dado e de controle, bem como, uma visão global da rede para prover solução de gerenciamento flexível e programável. Contudo, apesar do controle distribuído, a maioria das implementações ainda utiliza um único controlador SDN por domínio. Isto pode acarretar problemas de escalabilidade e falhas, sobretudo, em um ambiente dinâmico como o das VANETs, devido à enorme troca de informações de status entre o controlador e elementos do plano de dados, isto é, nós da rede, o que prejudica os requisitos de latência para aplicações de distribuição de conteúdo. De forma complementar, a Virtualização de Funções de Rede (Network Functions Virtualization - NFV), ainda não explorada usando a arquitetura MANO (Management and Orchestration) do NFV no contexto de gerenciamento de VANETs, permite configurar, controlar e a instanciação sob demanda de funções de rede, como (balanceamento de carga, roteamento, DHCP, NAT). Esta dissertação propõe e avalia uma solução flexível e dinâmica para o gerenciamento de redes veiculares através da sinergia entre os paradigmas SDN e NFV. A proposta concebe uma estratégia de auto escalonamento de controladores SDN configurados como Funções de Rede Virtualizadas (Virtualized Network Functions - VNFs) de acordo com as demandas de tráfego de cenários de redes veiculares. A proposta é implementada em um testbed de nuvem usando o serviço tacker baseado na arquitetura MANO do NFV integrada à plataforma Openstack. Os cenários de redes veiculares utilizam o emulador SDN para rede sem fio Mininet-WiFi, integrado ao simulador de mobilidade veicular SUMO que se baseia em um mapa da cidade. Os resultados obtidos demonstram a eficácia da proposta no suporte ao aumento de demanda de veículos conectados, redução na perda de pacotes, controle do jitter, e evita sobrecarga de processamento quando comparada ao desempenho da arquitetura tradicional.

**Palavras-chaves**: Redes Definidas por Software. Virtualização de Funções de Rede. Redes Veiculares.

# ABSTRACT

With the advent of Vehicular Ad Hoc Networks (VANETs), innovative services can be enabled to provide greater passenger comfort, accident prevention and improved mobility in large urban centers. In VANET communication called vehicle to infrastructure (V2I), the vehicle exchanges information with wireless access points arranged along the road. With the current architecture, some challenges in VANET require more research and new solutions due to the significant increase of connected devices, need for efficient use of resources, oscillation in information traffic, delays and packet losses due to the high mobility of vehicles, unreliable connections, and Quality of Service (QoS) requirements of the applications. To address the challenges of managing VANETs, has intensified the use of Software Defined Networking (SDN), which is based on the separation of data and control planes, as well as a network vision to provide flexible and programmable management solution. However, despite distributed control, most implementations still use a single SDN controller per domain. This can lead to problems of scalability and failures, especially in a dynamic environment such as VANETs, due to the huge exchange of status information between the controller and elements of the data plane, that is, network nodes, which impairs the requirements for content distribution applications. In addition, Network Functions Virtualization (NFV), not yet exploited using the MANO (Management and Orchestration) architecture of the NFV in the context of managing VANETs, allows configuration, control and on-demand instantiation of functions such as (load balancing, routing, DHCP, NAT). This dissertation proposes and evaluates a flexible and dynamic solution for the management of vehicular networks through the synergy between SDN and NFV paradigms. The proposal devises a strategy of self-scheduling of SDN controllers configured as Virtualized Network Functions (VNFs) according to the demands of traffic in vehicular networks. The proposal is implemented in a cloud testbed using the tacker service based on the NFV MANO architecture integrated to the Openstack platform. The network scenarios use the SDN emulator for wireless network called Mininet-WiFi, integrated to the simulator of vehicular mobility SUMO that is based on a map of the city. The results obtained demonstrate the effectiveness of the proposed solution to support the increasing demand of connected vehicles, reduction in packet loss, jitter control, and avoids processing overhead when compared to the performance of traditional architecture.

**Key-words**: Software Defined Networking. Network Function Virtualization. Veicular Ad Hoc Networks.

# LISTA DE ILUSTRAÇÕES

# LISTA DE TABELAS

# LISTA DE SIGLAS

| | |
|---|---|
| AAA | AUTHENTICATION, AUTHORIZATION AND ACCOUNTING. |
| API | APPLICATION PROGRAMMING INTERFACE. |
| CDN | CONTENT DELIVERY NETWORK. |
| DHCP | DYNAMIC HOST CONFIGURATION PROTOCOL. |
| DNS | DOMAIN NAME SYSTEM. |
| DPI | DEEP PACKET INSPECTION. |
| GPS | GLOBAL POSITIONING SYSTEM. |
| HSDV | HIERARCHICAL SOFTWARE-DEFINED VANET. |
| IoT | INTERNET OF THINGS. |
| ITS | INTELLIGENT TRANSPORTATION SYSTEMS. |
| MANET | MOBILE AD NETWORK. |
| MANO | MANAGEMENT AND ORCHESTRATION. |
| NAT | NETWORK ADDRESS TRANSLATION. |
| NFV | NETWORK FUNCTIONS VIRTUALIZATION. |
| NFVI | NETWORK FUNCTIONS VIRTUALIZATION INFRASTRUCTURE. |
| NFVO | NETWORK FUNCTIONS VIRTUALIZATION ORCHESTRATOR. |
| NSD | NETWORK SERVICE DESCRIPTORS. |
| OVNC | OVERLAY VEHICULAR NETWORK CREATION. |
| QoE | QUALITY OF EXPERIENCE. |
| QoS | QUALITY OF SERVICE. |
| RSU | ROAD SIDE UNIT. |
| SDN | SOFTWARE DEFINED NETWORKING. |
| SDVN | SOFTWARE-DEFINED VEHICULAR NETWORK. |
| SLA | SERVICE LEVEL AGREEMENT. |
| SUMO | SIMULATION OF URBAN MOBILITY. |
| TCP | TRANSMISSION CONTROL PROTOCOL. |
| TOSCA | TOPOLOGY AND ORCHESTRATION SPECIFICATION FOR CLOUD APPLICATIONS. |
| V2I | VEHICLE TO INFRASTRUCTURE. |
| V2V | VEHICLE-TO-VEHICLE. |

| | |
|---|---|
| VANET | VEICULAR AD HOC NETWORKS. |
| VIM | VIRTUALIZED INFRASTRUCTURE MANAGER. |
| VMs | VIRTUAL MACHINES. |
| VNF | VIRTUALIZED NETWORK FUNCTION. |
| VNFD | VIRTUALIZED NETWORK FUNCTION DESCRIPTORS. |
| VNFM | NETWORK FUNCTIONS VIRTUALIZATION MANAGER. |

# SUMÁRIO

# 1 INTRODUÇÃO

As redes veiculares (VEICULAR AD HOC NETWORKS (VANET)) vêm ganhando cada vez mais destaque nas pesquisas tanto da comunidade acadêmica quanto da indústria automobilística. VANETs são um tipo de redes móveis ad hoc (MOBILE AD NETWORK (MANET)), em que os nós da rede são veículos ao invés de dispositivos móveis. VANET é considerada um componente chave para os sistemas de transportes inteligentes (INTELLIGENT TRANSPORTATION SYSTEMS (ITS)). Dentre as características exclusivas dessas redes, estão inclusas mobilidade, densidade da rede, e rápida mudança topológica (BATRA; SINGH, 2017). Para fornecer aplicações, serviço de Internet confiável, compatibilidade com dispositivos pessoais, comunicação com a computação em nuvem, gerenciamento de tráfego de veículos, arquitetura atual de redes veiculares possui limitações relacionadas ao modo de rede *ad-hoc* (BORCOCI; AMBARUS; VOCHIN, 2017).

Com a arquitetura atual, alguns desafios em VANETs parecem ainda sem solução. São exemplos suportar o aumento significativo de dispositivos conectados, uso de recursos de forma eficiente, atrasos devido à alta mobilidade, conexões não confiáveis, e qualidade de serviço (QUALITY OF SERVICE (QoS))(TRUONG; LEE; GHAMRI-DOUDANE, 2015). O estado da arte em VANETs, até o momento, tem como objetivo mostrar soluções para as aplicações que necessitem de garantias de atrasos na rede. Entretenimento e notificação de congestionamento de tráfego são exemplos dessas aplicações. (RIZZO et al., 2016).

Fornecer conteúdos multimídia e aplicações para os usuários em ambientes com uma grande concentração de tráfego veicular e serviços requisitando e trocando informações torna a gerência do tráfego da rede um desafio. A infraestrutura atual de rede não está adequada para lidar com estas numerosas trocas de informações entre os veículos, o que pode gerar sobrecarga, gerar congestionamento e até mesmo comprometer a qualidade dos serviços. A topologia dinâmica das redes veiculares ainda apresenta desafios, como a mobilidade e os obstáculos que impedem a propagação de sinal. (REZENDE et al., 2014).

As redes definidas por software (SOFTWARE DEFINED NETWORKING (SDN)), surgem como uma das soluções para lidar com o desafio de tratar as limitações na infraestrutura de rede atual. Esta tecnologia adota a estratégia de usar uma arquitetura para separar o plano de dados (switches) do plano de controle (lógica da rede). Então, as decisões do plano de dados são gerenciáveis através do plano de controle e controladas por softwares. Em geral, o controle é centralizado e o estado da rede é determinado por um controlador SDN (KREUTZ et al., 2015a).

Seguindo o princípio de que novas soluções precisam ser construídas para suportar o acentuado tráfego de internet das coisas (INTERNET OF THINGS (IoT)), uma dessas soluções é o uso da tecnologia inovadora conhecida como virtualização de funções de

redes (NETWORK FUNCTIONS VIRTUALIZATION (NFV)), estas funções podem ser servidores de nomes (DOMAIN NAME SYSTEM (DNS)), serviços de distribuições de IPs (DYNAMIC HOST CONFIGURATION PROTOCOL (DHCP)), *firewall*, e roteamento. Surgindo com o objetivo de fornecer redes econômicas, estáveis e com a flexibilidade de serviços (NAOHISA et al., 2016). O NFV é o paradigma de virtualização de funções de rede, que permite configurar, controlar e gerenciar redes avançadas (KLJAIC; SKORPUT; AMIN, 2016). Um benefício para arquiteturas tradicionais de redes, para IoT onde os dispositivos não são projetados para suportar o alto nível de escalabilidade.

Com o objetivo de usar o NFV para criar funções que antes eram implantadas em hardwares dedicados e transformá-las em instância de software ou funções de redes virtuais, (por exemplo, *firewall*, autenticação, *Load Balance*, NAT, IDS), algumas funções de controle também podem ser virtualizadas e movidas para a nuvem como uma função de rede virtual (VIRTUALIZED NETWORK FUNCTION (VNF)). Portanto, é possível virtualizar as funções de controle do controlador SDN e implementá-las dinamicamente. (MUNOZ et al., 2015)

Embora os conceitos de SDN e virtualização de funções de redes NFV tenham alguns elementos em comum, os termos são usados de forma distinta, porém as tecnologias podem funcionar em conjunto. O SDN pode atuar como um gerenciador dinâmico dos fluxos de tráfegos da rede e o NFV para mover funções virtuais em um ambiente de nuvem, por exemplo. (NIKOS; KUIPERS; A., 2016).

Esta pesquisa propõe e avalia uma arquitetura para gerenciamento de tráfego em VANETs que trata de questões arquiteturais, para viabilizar a flexibilidade e dinamicidade na instanciação de controladores SDN como VNFs em função das demandas de tráfego e veículos da rede. Para isso, consideramos o emprego integrado das tecnologias SDN e NFV para virtualizar as funções do plano de controle. A proposta é implementada em um *testbed* de nuvem em conjunto com a ferramenta de simulação Mininet Wi-Fi, usada para criar os cenários de redes veiculares. Os resultados obtidos demonstram a eficácia da proposta no tratamento flexível e dinâmico da sobrecarga da rede com o aumento do tráfego veicular, em relação à redução na perda de pacotes, ao processamento e ao jitter em comparação com uma arquitetura tradicional.

## 1.1 Objetivos

### 1.1.1 Objetivo Geral

O principal objetivo deste trabalho é criar uma arquitetura flexível e com suporte ao gerenciamento dinâmico do tráfego de redes veiculares através do emprego de uma solução que se beneficia da sinergia entre SDN e NFV. Em conjunto com a flexibilidade de funções e serviços do NFV, a programabilidade da rede fornecida pelo SDN permite adicionar e

gerenciar novas funcionalidades, permitindo mudanças dinâmicas e elásticas através de controladores virtuais.

### 1.1.2 Objetivos Específicos

Propor uma arquitetura SDN/NFV para VANETs;

1. Propor uma arquitetura SDN/NFV para VANETs;
2. Elaborar uma estratégia de orquestração de controladores SDN como VNFs;
3. Integrar serviços da arquitetura NFV em uma infraestrutura de nuvem privada;
4. Desenvolver um serviço de monitoramento e alarme para cenário de redes veiculares;

## 1.2 Estrutura da dissertação

Esta dissertação está organizada da seguinte forma: No capítulo 2 são apresentados conceitos e definições fundamentais sobre Redes Definidas por Software, Internet das Coisas, Redes veiculares, Funções de Redes Virtuais, Computação em Nuvem e Escalabilidade. O Capítulo 3 ressalta e discute os trabalhos relacionados com o tema da pesquisa. Em seguida, o Capítulo 4 apresenta a proposta de arquitetura VSDN para VANETs. O Capítulo 5 apresenta os resultados obtidos por um conjunto de experimentos realizados considerando cenários e métricas. Finalmente, o Capítulo 6 conclui o trabalho apresentando um resumo dos resultados alcançados e mostrando direções para trabalhos futuros.

# 2 FUNDAMENTAÇÃO TEÓRICA

Esta seção abordará conceitos básicos para a realização desse trabalho. Na Seção 2.1 serão apresentados conceitos referentes às tecnologias SDN, VANETs e NFV demonstrando exemplos de suas arquiteturas. Em seguida na Seção 2.2 serão apresentados os elementos que formam o ambiente, funcionamento dos controladores OpenFlow, assim como as ferramentas utilizadas. Por último, a Seção 2.3 abordará os conceitos de computação em nuvem, que são fundamentais no desenvolvimento deste trabalho. Entre esses conceitos estão a elasticidade e os tipos de escalabilidade.

## 2.1 Tecnologias Constituintes da Proposta

### 2.1.1 VANETs

As redes veiculares são sistemas de comunicação que viabilizam a troca de informações entre veículos automotores e com infraestrutura fixa localizada às margens de ruas ou de estradas. As redes veiculares se diferenciam de outras redes sem fios pela sua mobilidade, por sua dinâmica e principalmente pela natureza dos nós, que são compostos por automóveis, caminhões e ônibus, com interfaces de comunicação sem fio, e por equipamentos fixos no entorno das vias (ALVES et al., 2009). As VANETs permitem uma comunicação direta entre os carros e com as bases fixas (ROAD SIDE UNIT (RSU)s). Assim, a situação do tráfego, e notificações de perigo nas vias podem ser enviadas e recebidas com o mínimo de latência possível (HARTENSTEIN; LABERTEAUX, 2008).

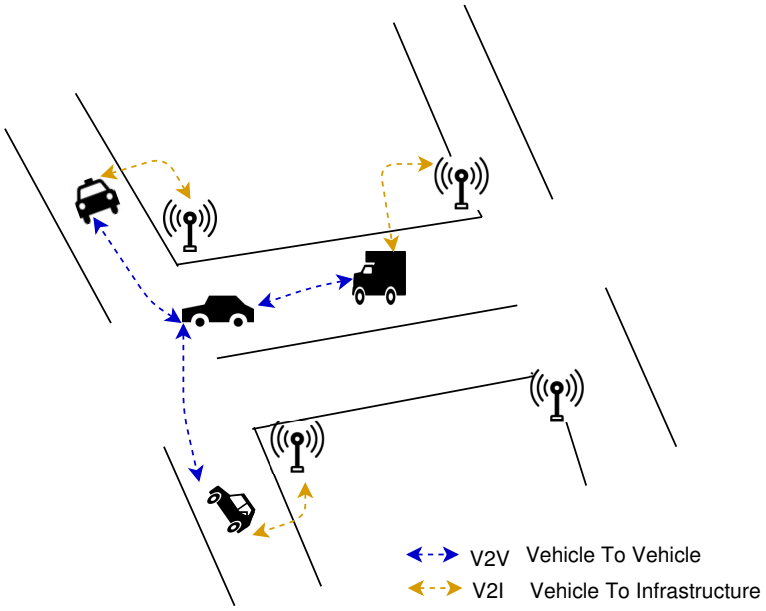
A proposta de usar veículos conectados traz uma variedade de serviços de sistemas de transportes inteligentes ITS que requerem conexões flexíveis, escaláveis e eficientes, entre os veículos e as RSUs. Além disso, serviços ITS podem ter requisitos para aplicações que necessitam de qualidade de serviço. Ainda no contexto VANET, é indispensável existir uma capacidade de lidar com a grande quantidade de requisições simultâneas, visto que o número de veículos conectados aumenta. (CHEN et al., 2017).

Em redes veiculares que compõem ITS, o veículo pode tanto receber quanto transmitir infromações. O ITS fornece três tipos de comunicação VANET (BHOI; KHILAR, 2014):

- Veículo para veículo (VEHICLE-TO-VEHICLE (V2V)) - Na comunicação, toda a comunicação é feita par a par, onde cada veículo pode atuar tanto como provedor de informações como consumidor, distribuindo as informações para outros veículos na rede. A transmissão de dados é feita através de *multi-hop* (*multicasting* /*broadcasting*).

- Veículo para infraestrutura (VEHICLE TO INFRASTRUCTURE (V2I)) – Nessa comunicação, a infraestrutura sem fio é usada para coletar informações de um veículo e fornecer a outros, se for necessário. Neste caso, as RSUs podem receber e enviar dados aos veículos, a comunicação usa apenas um salto e as mensagens são transmitidas apenas para os veículos que estejam ao seu alcance.
- Híbrido - A comunicação acontece das combinações do V2V e V2I. A união das duas comunicações tem como objetivo solucionar os eventuais problemas de cada arquitetura e somar seus benefícios. A Figura 1 demonstra uma visão lógica da comunicação em VANETs.

Figura 1 – Comunicação VANETs




**Fonte:** Elaborada pelo Autora

Uma das principais questões de pesquisa em redes veiculares está na concepção de uma arquitetura de sistema no qual seja possível usar várias tecnologias diferentes, por exemplo IEEE 802.11p, DSRC, WAVE, ITS G5, Wi-Fi ou 3G / 4G para fornecer uma rede veicular heterogênea. Além disso, para implementar esta heterogeneidade mencionada, os pesquisadores precisam da cooperação de diferentes parceiros e fabricantes para projetar uma arquitetura complexa e de grande escala. Assim, uma das principais tendências de pesquisa é desenvolver uma arquitetura de sistemas confiável, flexível e escalável. (LIANG et al., 2015).

#### *2.1.1.1 Aplicações VANETs*

Muitas aplicações veiculares podem ser desenvolvidas de acordo com os tipos de comunicação V2V e V2I. Assim, diferentes informações podem ser disseminadas para motoristas

e viajantes. A diversidade de dispositivos integrados ao veículo, como sensores, receptores de sistemas de localizações (GLOBAL POSITIONING SYSTEM (GPS)), concedem aos carros a capacidade de coletar, processar e compartilhar informações sobre si e sobre o ambiente, com outros veículos e com a infraestrutura (AL-SULTAN et al., 2013). Baseado no tipo de comunicação, as aplicações VANETs podem ser divididas nas seguintes categorias:

1. Aplicações de entretenimento: Esta categoria envolve as aplicações que não são relacionadas à segurança; Tendo como objetivo o conforto e o entretenimento do motorista, tornando a viagem mais agradável. São exemplos, informações meteorológicas, localização de restaurantes próximos ou lugares favoritos, acessar a Internet, *streaming* de vídeo e áudio, enviar e receber arquivos. Tudo isso é permitido enquanto o veículo está conectado à rede de infraestrutura.

2. Aplicações de segurança: Os aplicativos relacionados à segurança usam a comunicação sem fio V2V e V2I, com o intuito de melhorar e tornar as rodovias mais seguras, a intenção real é salvar vidas. As principais aplicações dessa categoria envolvem: alerta de engarrafamentos, alerta de emergência, aviso sobre velocidades das vias e aviso de rotas erradas. As notificações e soluções para prevenção de colisão veicular pode usar ambos os tipos de comunicação.

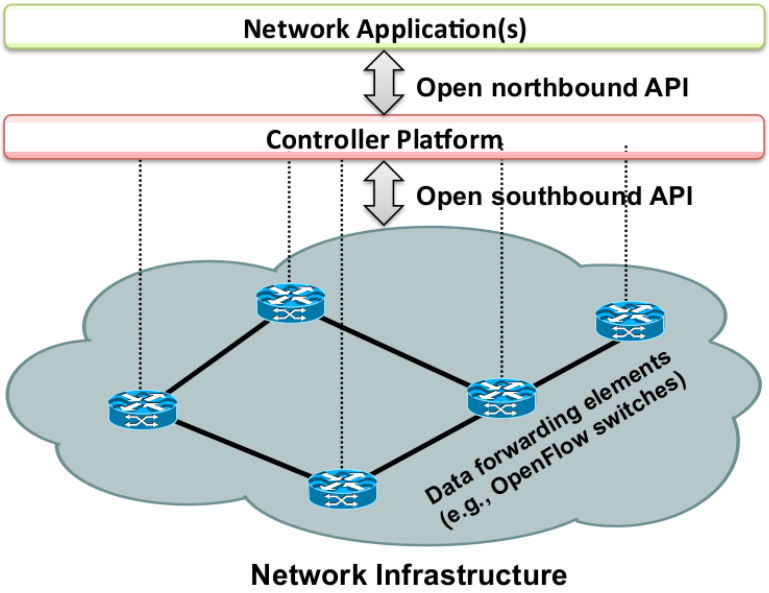
### 2.1.2 Software Defined Networking

A academia e a indústria têm voltado suas atenções significativas para as redes definidas por software. Na arquitetura SDN, os planos de dados e de controle estão desacoplados do hardware, o gerenciamento é programado no plano de controle e tem uma visão centralizada do estado lógico da infraestrutura.O protocolo *OpenFlow* elemento fundamental da arquitetura SDN, é usado para o controle lógico do nível dos fluxos na comunicação SDN.(YAN et al., 2016).

O SDN é um paradigma de rede emergente que pode mudar as limitações existentes na atual infraestrutura de redes. O plano de controle com a lógica de controle da rede são separados do plano de dados que possuem os roteadores e *switches* que encaminham o tráfego. Outro ponto é que, com esta separação dos planos, os *switches* tornam-se apenas dispositivos de encaminhamento e a lógica é implementada em um controlador centralizado ou sistema operacional de rede, tornando a aplicação de políticas e reconfiguração da rede mais simples (KREUTZ et al., 2015b). A Figura 2 mostra uma visão simplificada da arquitetura SDN.

A programação entre o plano de dados e o de controle, para realizar a separação, é feita usando uma interface de aplicação (APPLICATION PROGRAMMING INTERFACE (API)) definida entre os *switches* e o controlador SDN. O controlador exerce um controle direto sobre os elementos do plano de dados por meio da API de programação. (NIKOS; KUIPERS; A., 2016)

Figura 2 – Visão simplificada da Arquitetura SDN




**Fonte:** (KREUTZ et al., 2015b)

Dessa forma, é possível um melhor desempenho e configuração eficientes, tornando as redes flexíveis para acompanhar as tecnologias emergentes. (KIM; FEAMSTER, 2013) afirma que um controlador centralizado efetua o controle lógico da rede e transfere regras de encaminhamento para os *switches* usando o protocolo padronizado *openflow* , simplificando, dessa forma, a aplicação de políticas, a reconfiguração e a evolução da rede.

Em resumo, a principal ideia por trás dos conceitos e implementação do SDN é fazer com que as decisões relevantes sejam tomadas pelo plano de controle, com uma interface de programação que permite um software monitorar e alterar a rede. Após a separação, o plano de dados executa apenas o encaminhamento de pacotes para o destino mais adequado(NIKOS; KUIPERS; A., 2016) .

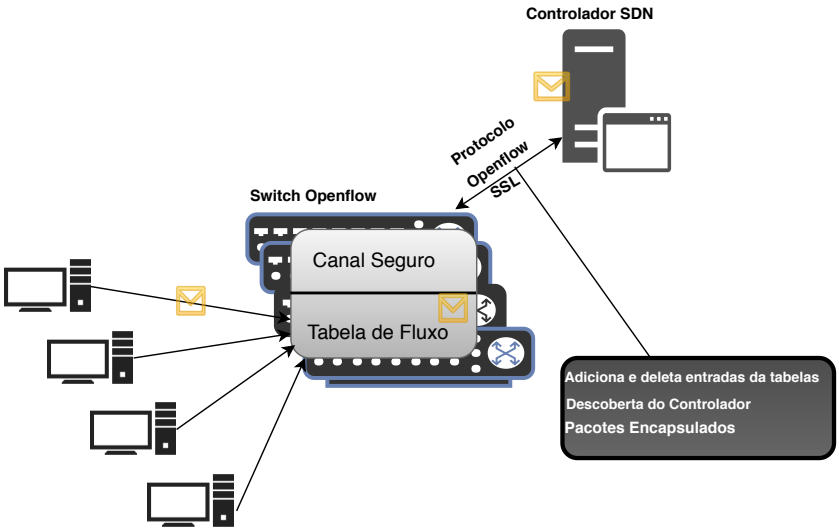
### 2.1.3 OpenFlow

O protocolo *OpenFlow* é um padrão SDN com um conjunto de mensagens de controle usadas para a comunicação entre planos de dados (*switch*) e plano de controle (controlador). Estas mensagens de controle podem ser organizadas no gerenciamento, o que inclui um controle do plano de dados e até mesmo monitoramento (LI; YOO; HONG, 2015). Uma extensão natural para que o roteamento seja controlado de forma centralizada pela rede é a padronização de protocolos SDN, como o *OpenFlow*.

Com isso, é permitida a implantação rápida de novos aplicativos e serviços destinados a reduzir custos ou aumentar a segurança, estabilidade ou disponibilidade em redes. O *OpenFlow* possui um canal seguro e usa controle de transmissão (TRANSMISSION

CONTROL PROTOCOL (TCP)) como protocolo de transporte para a comunicação até o controlador. Os *switches*, possuem uma tabela de fluxo contendo entradas e ações; e um protocolo para comunicação entre o controlador e o *switch*, além de um processo para gerenciar as novas entradas da tabela de fluxo. As redes SDN infraestruturadas, com um plano de controle centralizado, usam o *OpenFlow* para essa comunicação até o plano de dados, como aquelas encontradas em *data centers*. (MENDONCA et al., 2012). A figura 3 ilustra o funcionamento de um *switch openflow*.

Figura 3 – Switch OpenFlow




**Fonte:** Elaborada pela autora

As mensagens de controle para gerenciar o fluxo são definidas em quatro tipos: *Packet-In*, *Flow-Mod*, *Packet-Out* e *Flow-Removed*. Se nenhuma regra de fluxo estiver disponível para combinar os pacotes recebidos, uma mensagem *Packet-In* é iniciada automaticamente pelo protocolo *OpenFlow* e é enviada ao controlador solicitando qual o caminho a ser direcionada (LI; YOO; HONG, 2015).

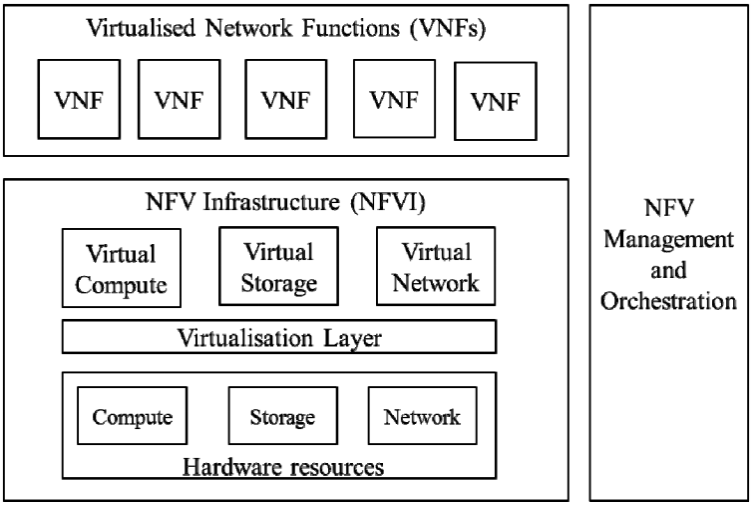
Os *switches* enviam mensagens *Packet-In* do tipo assíncrona para o controlador quando não existe nenhuma regra para um determinado pacote. Ao receber essa mensagem, uma alteração é realizada para o encaminhamento do destino resultando em uma mensagem *Flow-Mod*, o estado do *switch openflow* é modificado por ela. Em seguida, o controlador envia uma mensagem *Packet-Out* informando que os pacotes estão prontos para serem encaminhados. Como o espaço para armazenar as regras é limitado, quando existe alguma regra de fluxo ociosa, elas precisam ser removidas. Portanto, um *Flow-Removed* é aplicado para remover uma entrada de fluxo de uma tabela (ONF, 2013).

### 2.1.4 Network Functions Virtualization

Virtualização de funções de rede NFV é uma tecnologia que virtualiza as funções de rede, que antes eram implementadas usando hardware dedicado, e agora essas funções podem ser virtualizadas como softwares, e executadas em um único servidor. Também é possível estender, alterar ou utilizar recursos virtuais dos equipamentos de acordo com as necessidades (NAOHISA et al., 2016).

O NFV representa um avanço para diversos serviços em ambientes de telecomunicações e redes. Desacoplar o *software* do *hardware*, implantação de funções de redes flexíveis e operações dinâmicas são algumas das diferentes maneiras de provisionamento de rede. A Figura 4 ilustra a arquitetura NFV de alto nível definida pela ETSI com os três principais domínios de trabalho descritos pelo (ETSI, 2013).

Figura 4 – Arquitetura NFV ETSI




**Fonte:** (ETSI, 2013)

Na arquitetura da Figura 4 a função de rede virtual VNF é a implementação de um software capaz de executar na camada de infraestrutura do NFV chamada de infraestrutura do NFV (NETWORK FUNCTIONS VIRTUALIZATION INFRASTRUCTURE (NFVI)). O NFVI é uma camada de infraestrutura que inclui diversos recursos físicos do hardware como armazenamento e rede, e também contém uma camada de virtualização que define como eles podem ser virtualizados. O NFVI tem suporte a execução das VNFs .

A gerência e orquestração (MANAGEMENT AND ORCHESTRATION (MANO)) abrange o gerenciamento do NFVI com o ciclo de vida dos recursos físicos e de software na infraestrutura de virtualização e no ciclo de vida das VNFs. Na arquitetura MANO existem blocos funcionais para orquestrar a rede, gerenciar as VNFs e a infraestrutura de virtualização. O MANO tem como objetivo gerenciar todas as tarefas necessárias para virtualização no *framework* NFV permitindo uma integração ágil e uma gestão adequada.

As VNFs podem ser implementadas e distribuídas para o compartilhamento de diferentes recursos físicos ou virtuais de infraestrutura, provendo os requisitos de escalabilidade e desempenho, tornando mais fácil a implantação de novos serviços. O NFV pode ser aplicado a qualquer processamento de pacote, em plano de dados e qualquer uma de suas funções, no plano de controle, tanto em redes móveis quanto em redes fixas. Alguns exemplos de funções de redes a serem virtualizadas incluem:

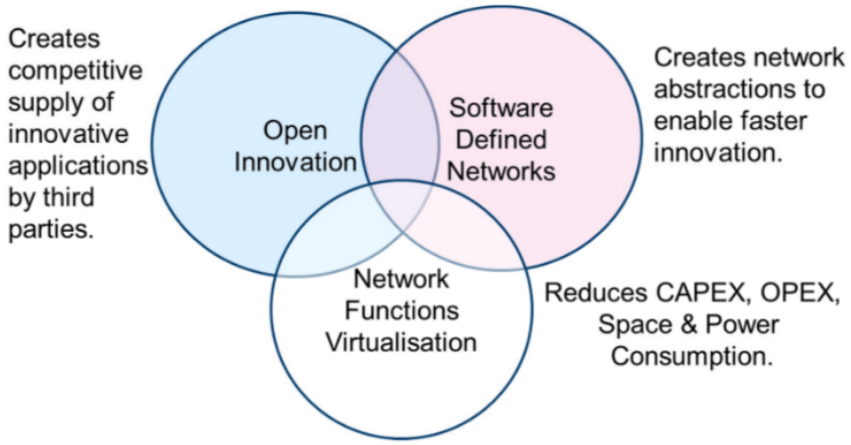
- Elementos de comutação - Tradução de endereços de redes (NETWORK ADDRESS TRANSLATION (NAT)) e roteadores .
- Análise de tráfego - medição, inspeção de pacotes (DEEP PACKET INSPECTION (DPI)) e qualidade da experiência (QUALITY OF EXPERIENCE (QoE))
- Garantia de serviço - Acompanhamento de garantia de nível de serviço (SERVICE LEVEL AGREEMENT (SLA)), testes e diagnósticos.
- Funções de redes convergentes - Servidores de autorização e autenticação (AUTHENTICATION, AUTHORIZATION AND ACCOUNTING (AAA)), políticas de controle e plataformas e carregamento.
- Otimização no nível de aplicação - Redes centradas em conteúdos (CONTENT DELIVERY NETWORK (CDN)), servidores de cache, balanceadores de carga.
- Recurso de segurança - *firewalls*, antivírus, sistemas de detecção de intrusão, proteção contra spam. (HERRERA; VEGA, 2016)

O NFV pode ser usado baseado em SDN, e na nuvem virtualizar as funções conhecidas como VNF, para tornar a rede flexível e programável desacoplando as funções de redes do hardware. Assim, o papel da nuvem no ambiente VANET é fundamental para controlar as RSUs em suas áreas, e é uma maneira de reduzir custos de implementação (CHEN et al., 2017). Também é possível virtualizar funções de um plano de controle SDN, por exemplo, um controlador SDN e movê-lo dinamicamente na nuvem como instâncias, isso em poucos minutos(MUñOZ et al., 2015).

O NFV vem para complementar a idealização do SDN, a combinação das duas tecnologias oferece uma maior integração para as redes atuais. Além do mútuo benefício, elas partilham da mesma característica de promover a inovação e criatividade. As duas soluções podem ser usadas em conjunto para um maior aproveitamento, por exemplo, conectar arquiteturas SDN pode dar suporte ao NFV para melhorar seu desempenho e facilitar seu funcionamento. Porém, é importante deixar claro que a virtualização e a implantação de funções de redes não dependem das tecnologias SDN, assim como o SDN também não depende do NFV(HAN et al., 2015).

A Figura 5 ilustra a relação entre as tecnologias SDN e NFV:

Figura 5 – Integração NFV com SDN




**Fonte:** (HERRERA; VEGA, 2016)

Conforme ilustrado na Figura 5, o NFV é altamente complementar a SDN. Com a união das tecnologias, as abordagens propostas na separação do plano de controle e de dados do SDN podem melhorar o desempenho, operação e manutenção de infraestrutura. O NFV, por sua vez, reduz os custos de despesas computacionais e de capital, além dos gastos relacionados ao espaço e energia, assim, como é afirmado na parte azul da imagem, surgem ofertas de aplicações inovadores (HERRERA; VEGA, 2016).

## 2.2 Ferramentas e Ambientes de Avaliação

### 2.2.1 Mininet WiFi

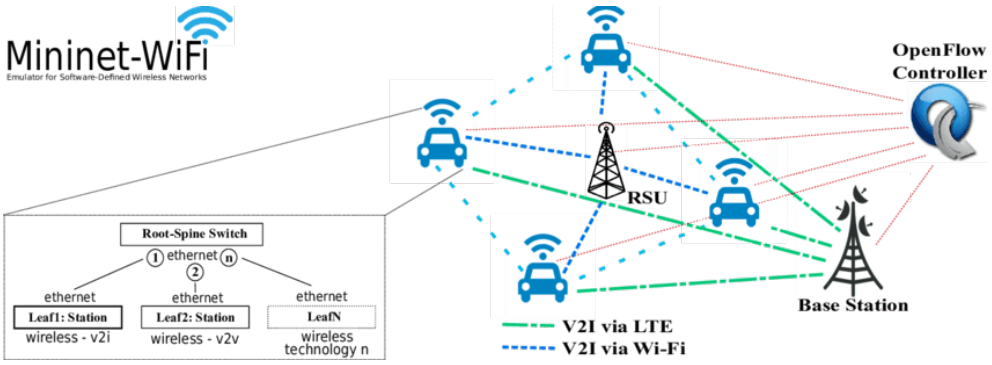
O Mininet-WiFi [1] é um software de código aberto, desenvolvido utilizando a linguagem python, é uma extensão do emulador de redes *Mininet*, que permite o uso de estações WiFi e pontos de acesso. A plataforma permite emular cenários SDN e acrescenta recursos das redes sem fios, os recursos originais do emulador *Mininet*, como computadores e controladores *OpenFlow*, ainda são mantidos. Para permitir o suporte à tecnologia WiFi, as estações (STAs) e o pontos de acesso (APs) são virtualizadas através do *driver* sem fio do Linux mac80211/SoftMac [2] (FONTES et al., 2015).

O emulador possui uma implementação de uma arquitetura *node car*, permitindo cenários de VANETs realísticos. As comunicações V2I, entre os carros e as RSUs ou estações bases *eNodeBs*, são estabelecidas no modo sem fio V2I, onde cada nó hospedeiro fornece interfaces sem fio. Já as comunicações V2V entre os carros, são estabelecidas no modo sem fio usando outras interfaces sem fio. Além disso, a arquitetura ainda permite o uso da

[1] https://github.com/intrig-unicamp/mininet-wifi
[2] http://linuxwireless.org/en/developers/Documentation/mac80211

programação SDN, usando todas as interfaces ao mesmo tempo, sendo possível escolher qualquer controlador *OpenFlow*, seja centralizado ou distribuído(FONTES et al., 2017) . A Figura 6 apresenta a arquitetura do Mininet-WiFi para cenários de redes veiculares.

Figura 6 – Arquitetura do Mininet-WiFi para VANETs




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

Nesta arquitetura, podemos observar que são necessárias três interfaces veiculares para que os carros usem canais de comunicações diferentes. No caso, uma interface para as comunicações veículo com veículo V2V, outra para comunicação com a RSU V2I e outra para interações com estações bases (sem fio).

### 2.2.2 Simulador de Mobilidade - SUMO

O simulador de mobilidade urbana, SIMULATION OF URBAN MOBILITY (SUMO), é escrito em C++ e destinado ao planejamento de otimização de projetos rodoviários. A ferramenta possui um conjunto de simulação de tráfego gratuito, com mapas configurados das cidades. Além disso, também é possível modelar sistemas de tráfego intermodal, ou seja, várias modalidades de transportes, veículos rodoviários, transportes públicos e pedestres (DLR, 2017).

A documentação da ferramenta ainda apresenta funcionalidades executadas pelo simulador, como avaliação de desempenho e o funcionamento de semáforos, avaliação de roteamento veicular baseado na emissão de gases poluentes. O SUMO é amplamente utilizado pela comunidade de pesquisa em VANET fornecendo *traces* realísticos de veículos, a ferramenta também fornece suporte à integração com simuladores de redes. O modelo de mobilidade adotado pelo SUMO é o microscópico (modelagem de veículos), algumas das funções realizadas pelo simulador consistem em:

- Prover uma simulação microscópica - onde veículos, pedestres e transportes públicos são modelados explicitamente;
- Simulação de tráfego multimodal, por exemplo, veículos, transportes públicos e pedestres;

- No SUMO ainda é possível importar ou gerar os horários dos semáforos automaticamente;
- Não existem limites para o tamanho da rede ou dos veículos; e
- Tem suporte aos formatos de importação de mapas: *OpenStreetMap*[3], *VISUM*[4], *VISSIM*[5], *NavTeq* [6].

Os mapas são convertidos e integrados com os recursos de simulação do SUMO, então o simulador de rede gera as análises no nível de rede e é integrado aos mapas gerados e aos parâmetros de mobilidade (DLR, 2017).

#### 2.2.2.1 Mobilidade dos Nós

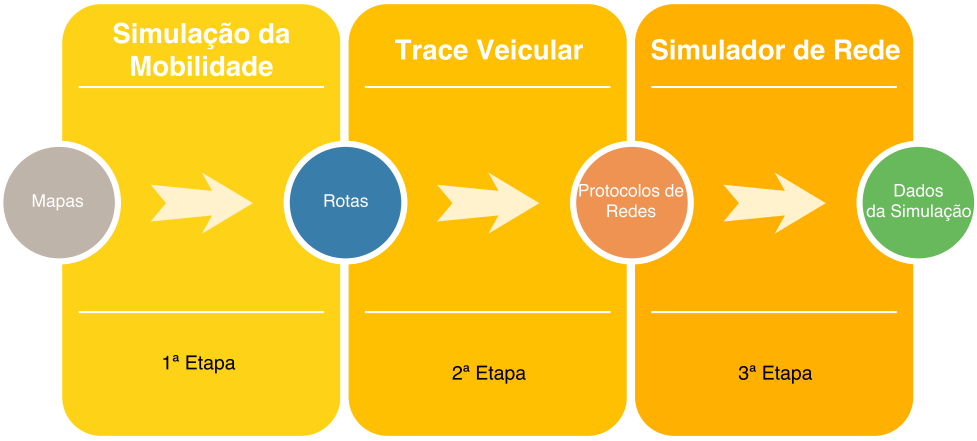
Nas redes veiculares, a movimentação dos veículos não é aleatória, a topologia fixa com prédios e ruas define os limites e veículos mais próximos e restringem os movimentos(AL-SULTAN et al., 2013). Para avaliar protocolos e soluções em redes veiculares muitas vezes são necessário realizar simulações, visto que realizar testes em um ambiente real tem um custo maior. Para que estas simulações sejam o mais próximo possível ao ambiente real, estudos utilizam modelos de mobilidade para representar a movimentação dos veículos. Nesta seção serão apresentados os modelos de mobilidade utilizados em redes veiculares, e a ferramenta usada neste trabalho para prover um modelo de mobilidade nas simulações.

Para simular uma rede VANET pode ser necessário utilizar dois componentes diferentes: um simulador de tráfego veicular, que seja capaz de prover um modelo de mobilidade para os nós VANET e um simulador de rede para simular o comportamento de uma rede sem fio. Na prática, é necessário utilizar um software que gere os movimentos dos carros através de um arquivo de *trace*, e o simulador de rede para testar o desempenho dos protocolos de rede e um possível *middleware* caso necessário, para integrar o simulador de tráfego e o simulador de rede (SPAHO et al., 2011). A Figura 7 ilustra um esquema utilizado para simulação de redes veiculares dividido em três partes. A primeira parte consiste em usar um simulador de tráfego, a segunda parte é a geração dos traces da mobilidade e a terceira parte o simulador de rede é usado.

Os nós das redes veiculares, diferente de outras redes móveis, não se movimentam aleatoriamente. Então, é necessário definir o mapa a ser usado no simulador de mobilidade, criando a topologia de rede para as restrições de movimentação dos carros e definindo as rotas. Então, para que os veículos possam interagir entre si e com a topologia de rede é preciso usar um modelo de mobilidade. A seguir, serão apresentados a classificação dos modelos.

[3] https://www.openstreetmap.org/
[4] http://vision-traffic.ptvgroup.com/nl/products/ptv-visum/
[5] http://vision-traffic.ptvgroup.com/nl/products/ptv-vissim/
[6] https://mapupdate.navigation.com/

Figura 7 – Esquema de Simulação em Redes Veiculares




**Fonte:** Elaborada pelo Autora

Várias abordagens foram adotadas na modelagem da mobilidade seguindo a classificação baseada no nível da representação dos movimentos de acordo com (OLARIU; WEIGLE, 2009) os modelos são divididos nas seguintes categorias:

- **Modelo Macroscópico:** Neste modelo apenas as características relacionadas à velocidade e à densidade são levadas em consideração para a modelagem, e todo tráfego veicular é um fluxo contínuo. Além disso, neste modelo não é considerada a mobilidade de um carro individual.
- **Modelo Microscópico:** Para este modelo cada veículo tem sua movimentação e dinâmica representada em detalhes, tratada independente dos outros veículos. Exceto para carros que estão em uma proximidade que pode impactar o comportamento do motorista. Estes modelos são considerados de alta complexidade computacional por conseguirem reproduzir algumas situações do mundo real, como mudança de faixa e comunicação com o veículo da frente.
- **Modelo Mesoscópico:** Com este modelo é possível modelar características individuais dos veículos, como a densidade relacionada a métricas macroscópicas. O modelo fica entre os modelos microscópico e macroscópico. De forma resumida, a ideia é ter o benefício da escalabilidade macroscópica e permanecer com detalhes que estão presentes apenas nos modelos microscópicos.

### 2.2.3 Controlador SDN - Ryu

Na arquitetura SDN, os controladores são as entidades centrais que oferecem a capacidade dos clientes (aplicativos, usuários e inquilinos) de interagir com o gerecimento de controle. Isso ocorre através das chamadas interfaces *Northbound*(ONFTR-526, 2016).

O *Ryu* é um controlador de redes definida por software com código aberto e totalmente escrito em *python*. A *API* do software facilita o desenvolvimento de novos aplicativos para gerenciamento. Para gerenciar os dispositivos da rede, o *ryu* suporta o protocolo *OpenFlow* nas suas principais versões (OSRG, 2017). Nas redes definidas por software, existem inúmeros controladores: *POX*, *NOX*,*Maestro*, *FloodLight*, *Beacon*, *OpenDaylight*, *ONOS*, *Beacon*, *Onix*, *Mul*, *Ryu*, *Trema*, etc (KHAN et al., 2016).

Como o controle é centralizado, um controlador SDN é responsável por gerenciar e implementar as mudanças na rede. Os controladores *OpenFlow* além da função de monitorar a rede, também comanda seus dispositivos. Dessa forma, o controlador tem acesso a esses dispositivos através de um canal seguro (JAIN; PAUL, 2013).

### 2.2.4 Openstack e Tacker

O *openstack* é um sistema operacional de nuvem com código aberto e escrito em *python*. Fornece uma infraestrutura como serviço (IaaS), além dos recursos de nuvem como armazenamento e a virtualização, pode gerenciar vários *hypervisors*, responsáveis por fornecerem abstração das máquinas virtuais (VIRTUAL MACHINES (VMs)). Os serviços são projetados para fornecerem uma escalabilidade grande (SAHASRABUDHE; SONAWANI, 2014).

A plataforma suporta todos os tipos de ambientes de computação em nuvem. A arquitetura funcional é construída com um nó de rede para controle e gerenciamento dos serviços *controller* e um de computação para armazenamento e provisionamento das máquinas virtuais. Dentre os principais serviços e projetos necessários para o bom funcionamento do *openstack* estão o *nova* para gerenciar o ciclo de vida das VMs, o *neutron* para conectividade e suporte às tecnologias de rede. Para os serviços de monitoramento e telemetria é usado o *ceilometer* para fins de estatísticas e escalabilidade (OPENSTACK, 2018).

O *Openstack* ainda possui projetos como soluções distribuídas, nuvens privadas e públicas, envolvendo orquestração, ciclo de vida de aplicações, serviços de rede, armazenamento, ferramentas de monitoramento, compartilhamento de arquivos, infraestrutura de *container* computacional e NFV (OPENSTACK, 2018).

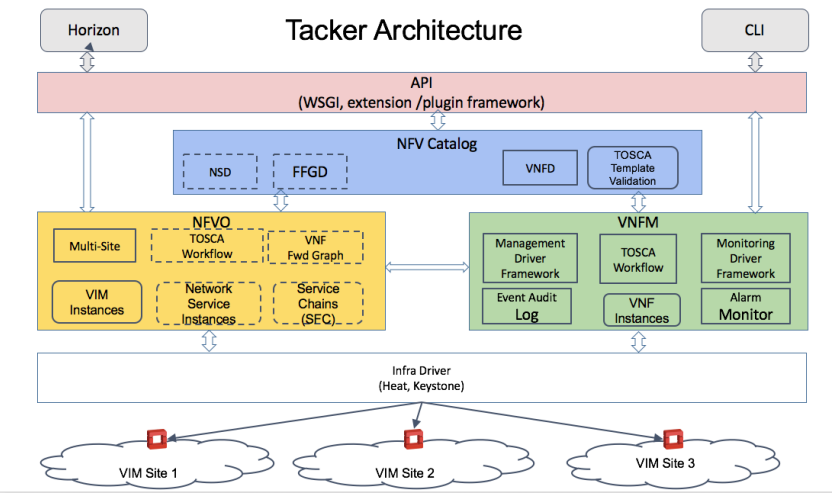
#### *2.2.4.1 Serviços NFV*

Nesta subseção, apresentaremos alguns dos principais projetos projetos NFV como *OpenMANO* e *OPENFV* que estão sendo desenvolvidos globalmente pela indústria e na comunidade acadêmica. Em destaque para o projeto do *tacker*, desenvolvido pela comunidade *openstack* e que foi usado neste trabalho.

1. **Tacker** - O *Tacker* é um projeto do *openstack* baseado na arquitetura ETSI MANO funciona como um orquestrador NFV administrando todas as VNFs e serviços de

rede. No sistema *Tacker MANO*, a VNF pode ser instalada em um *openstack* de destino que também é chamado de gerenciamento da infraestrutura virtualizada (VIRTUALIZED INFRASTRUCTURE MANAGER (VIM)) (TACKER, 2016). A arquitetura de alto nível do *tacker* é representada na Figura 8. Os componentes são descritos conforme a definição na documentação oficial do projeto.

Figura 8 – Arquitetura *Tacker* de Alto Nível




**Fonte:** (TACKER, 2016)

- VNF *Catalog* - Descrições de serviços de rede de VNFs através dos padrões de topologia e orquestração para aplicações de nuvem (TOPOLOGY AND ORCHESTRATION SPECIFICATION FOR CLOUD APPLICATIONS (TOSCA)), e de gráfico de encaminhamento das VNFs.

- O gerenciador das funções de redes virtuais (NETWORK FUNCTIONS VIRTUALIZATION MANAGER (VNFM)) - VNF *Manager* provê o ciclo básico de vida das VNFs, monitoramento do comportamento das VNFs, políticas de escalonamento, facilita a configuração das VNFs. O VNFM trabalha em conjunto com outros componentes do NFV, como o orquestrador NFV (NETWORK FUNCTIONS VIRTUALIZATION ORCHESTRATOR (NFVO)) e o VIM, o que permite padronizar as VNFs.

- NFVO - O orquestrador é responsável por prover o *template* com os serviços de rede fim a fim, políticas de instanciação das NFVs, verificação e alocação de recursos do VIM, capacidade de orquestrar VNFs em vários VIMs. O NFVO funciona como um orquestrador central para as unir funções de redes e coordenar os serviços, em ambientes NFV

Todos esses componentes, apresentados de acordo com a arquitetura do *tacker*, podem reduzir custos e aumentar a velocidade das implantações de serviços. Para os cenários das redes virtualizadas do NFV gerenciar e orquestrar ainda são desafios, quando se trata de escalabilidade e velocidade.

(2) **OpenMano** - É um projeto de código aberto desenvolvido para permitir uma implementação prática do padrão NFV ISG MANO da arquitetura de referência para gerenciamento e orquestração. Consiste de três componentes:

- OpenMANO - É um componente implementado de acordo com a referência de um NFVO. Este componente é responsável pela interação do NFV com o VIM através de sua API de programação, que permite oferecer serviços de criar e excluir templates VNFs, instanciar VNF e criar modelos de serviços.
- OpenVIM - Implementa o padrão VIM do NFV interagindo com nós de computação e a infraestrutura do NFV, e possue um controlador para fornecer os recursos de rede e computação necessárias à criação de VNFs. Sua API oferece serviços de nuvens, como criar, excluir e gerenciar imagens dos sistemas operacionais, a configuração de hardware disponível para as VNFs, ou seja, *flavors*, além das instâncias ou VMs e redes.
- OpenMANO-gui - Interface gráfica que permite ao usuário interagir com a API *OpenMANO* de forma simplificada (OPENMANO, 2014).

(3) **OPNFV** - A fundação Linux é responsável pela criação da plataforma aberta que foi anunciada em 2014 para criar projetos NFV para operadoras e fornecedores de serviços, além do desenvolvimento e a evolução dos componentes NFV. Os principais objetivos do projeto são:

- Desenvolver e investigar as funcionalidades do NFV.
- Validar se a plataforma satisfaz as necessidades do NFV, como orquestração e gerenciamento .
- Contribuir e participar de projetos de código aberto envolvendo NFV.
- Estabelecer soluções NFV com base em padrões e software livres.
- Promover o OPNFV como plataforma de referência para aplicações NFV.

A solução OPNFV envolve NFVI + VIM viabilizando uma especificação única para APIs que permite que um componente da rede se comunique com um componente de alto nível ou de nível inferior (OPNFV, 2014).

## 2.3 Escalabilidade em computação em nuvem

A computação em nuvem é uma tecnologia já consolidada para serviços de informação sob demanda. Uma nuvem é definida como um sistema distribuído que trabalha de forma paralela, formada por computadores conectados e virtualizados. Baseado no nível do serviços, os recursos podem ser provisionados dinamicamente através de acordos entre o prestador de serviço e o consumidor (BUYYA et al., 2009).

### 2.3.1 Escalabilidade e Elasticidade

É comum o termo elasticidade ser usado como sinônimo de escalabilidade, mas na verdade eles são conceitos diferentes e não devem ser confundidos ou usados da mesma forma. A escalabilidade é a capacidade de um sistema ser ampliado para um tamanho que deveria suportar crescimentos futuros, ou ter a capacidade de melhorar de acordo com a proporção de recursos adicionados. Diz-se que uma nuvem é escalável quando é possível adicionar recursos sempre que ocorre um aumento de demanda, para manter as aplicações disponíveis ao nível requerido (GALANTE; BONA, 2012).

Elasticidade é a capacidade de um sistema ou nuvem prover, de forma dinâmica e otimizada, recursos computacionais de acordo com o aumento ou diminuição da carga de trabalho. Uma das características mais inovadoras da computação em nuvem, a elasticidade pode provisionar e desprovisionar os recursos em tempo de execução de forma à acompanhar a demanda. Por outro lado, a escalabilidade é usada apenas para adicionar novos recursos, garantido que mesmo com o aumento do custo operacional, o sistema será escalável. Quando se escala um sistema não há preocupação com a remoção de recursos ociosos (LEHRIG; EIKERLING; BECKER, 2015).

A elasticidade torna a computação em nuvem dinâmica. Existem dois tipos de escalabilidade: a horizontal e a vertical, as Seções 2.3.2 e 2.3.3 apresentarão o funcionamento e a diferença entre as duas.

### 2.3.2 Escalabilidade Horizontal

Escalabilidade horizontal - conhecida também como replicação ou *scale out/in*, consiste em adicionar ou remover instâncias (máquinas virtuais) e também contêineres (sistemas operacionais executando como processos) em ambientes virtualizados. Na maioria dos provedores públicos e em muitos trabalhos a replicação é atualmente o método mais utilizado para fornecer escalabilidade e elasticidade (GALANTE; BONA, 2012).

A escalabilidade horizontal é mais utilizada e mais simples de ser implementada suportando a maioria dos *hypervisores* (camadas de virtualização). Nessa abordagem, as VMs são fornecidas de forma estática, o que torna a utilização dos recursos ineficiente. Geralmente, balanceadores de cargas são usados para distribuir a carga entre as instân-

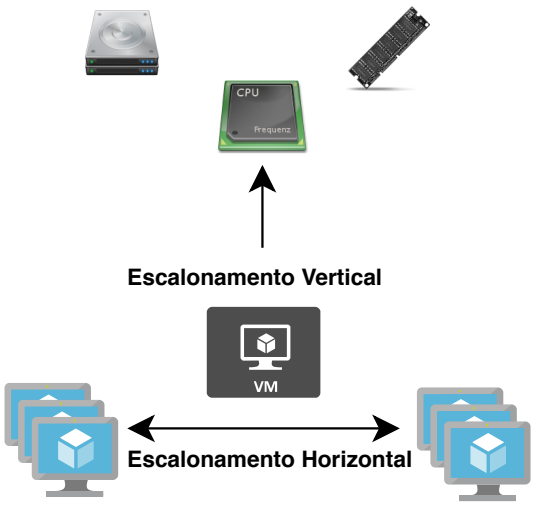
cias, que podem estar localizadas em lugares diferentes, prática comum na migração entre nuvens. O balanceamento é o método mais utilizado pelo provedor de nuvem pública EC2 da *Amazon* (AL-DHURAIBI et al., 2018).

### 2.3.3 Escalabilidade Vertical

Escalabilidade vertical também chamada *scale up/down* consiste em um método de redimensionamento, ou seja, é possível adicionar ou remover recursos de processamento, memória, CPUs e armazenamento de uma instância virtual (GALANTE; BONA, 2012). Neste método, os recursos são modificados enquanto as VMs estão em execução, sendo necessárias técnicas e mecanismos para redimensionar CPUs, discos entre outros (AL-DHURAIBI et al., 2018).

As ações de escalonamento, ao utilizar a API *tacker*, incluem dimensionar, criar e remover VNFs sob demanda, realizando de formar manual (*scaling*) ou automática (*auto-scaling*)[7] . Para diferenciar os dois métodos de redimensionamento a Figura 9 ilustra a diferença entre a técnica horizontal e a vertical.

Figura 9 – Tipos de Escalonamentos




**Fonte:** Elaborada pelo autora

A Figura 9 mostra que quando ocorre o dimensionamento horizontal mais máquinas virtuais são adicionadas como recurso. No entanto, no escalonamento vertical, adiciona-se mais poder a uma máquina existente como CPU, memória RAM. De uma maneira simples, percebe-se que a VMs na cor preta, adiciona mais máquinas da direção horizontal e no caso dos recursos são acrescentados no sentido vertical.

[7] https://specs.openstack.org/openstack/tacker-specs/specs/newton/manual-and-auto-scaling.html

## 2.4 Considerações Finais

Este capítulo apresentou um resumo dos principais conceitos que fundamentam a criação da arquitetura que será apresentado nessa proposta. Inicialmente foram apresentados os principais conceitos relativos à as tecnologias constituintes da proposta, seguido dos conceitos que envolvem as ferramentas e o ambiente possibilitando ao leitor conhecer de forma básica os conceitos que fundamentam esse trabalho. O Capítulo 3 apresentará um resumo de alguns trabalhos relacionados aos temas expostos neste Capítulo.

# 3 TRABALHOS RELACIONADOS

Este capítulo apresenta os trabalhos relacionados à integração de SDN e NFV para uso em VANETs . Ele está dividido em três subseções. A Seção 3.1 aborda artigos sobre arquiteturas para gerenciamento de VANETs usando SDN. A Subseção 3.2 mostra alguns trabalhos relacionados ao NFV em redes veiculares. Na Subseção 3.3 são discutidos os trabalhos que abordam arquiteturas usando as duas tecnologias SDN e NFV para VANETs. A discussão a respeito das comparações e contribuições dos trabalhos são realizadas na Subseção 3.4. Por fim, a Subseção 3.5, apresenta as considerações finais deste capítulo.

## 3.1 Gerenciamento em Redes Veiculares com SDN

Alguns trabalhos motivam o uso de redes definidas por software para gerenciamento de redes veiculares, a fim de garantir flexibilidade e escalabilidade. As arquiteturas propostas vão desde a adaptação para atender às requisições de dispositivos IoT, até replicação de informações para nuvens privadas.

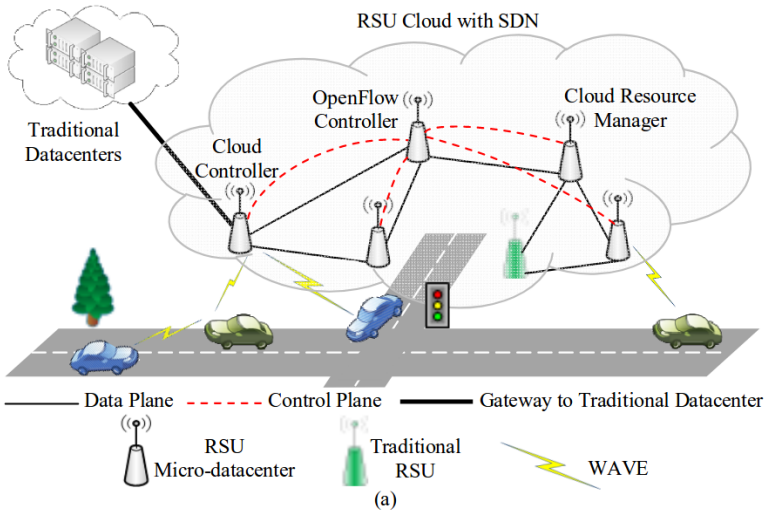
### 3.1.1 Redes Veiculares Definidas por Software

Na pesquisa de (SALAHUDDIN et al., 2014), os autores criam uma arquitetura SDN para gerenciar as redes veiculares usando computação em nuvem. Denominada nuvem de RSUs, o objetivo é garantir qualidade de serviço em aplicações de sistemas de transportes inteligentes que não sejam relacionados à segurança pessoal. Na nuvem, as RSUs podem ser reconfigurados, para atender às exigências de serviços dinâmicos. O SDN foi usado para migrações de máquinas virtuais e modificações do plano de controle. Como as reconfigurações frequentes na RSU afetam o desempenho da rede e a QoS dos aplicativos e serviços do ITS, a proposta utiliza uma técnica para gerenciar os recursos da RSU na nuvem, para minimizar o efeito de reconfiguração, a sobrecarga do plano de controle e o atraso na infraestrutura.

A arquitetura nuvem de RSUs proposta por (SALAHUDDIN et al., 2014) soluciona o alto custo das abordagens dos serviços de hospedagem em micro *data centers*, uma vez que os provedores de serviços são responsáveis por alugar recursos da infraestrutura de nuvem. A principal contribuição do trabalho é criar um gerenciador de recursos de nuvem *offline*, que é responsável na nuvem RSU por tomar as melhores decisões sobre a localização e o número de serviços hospedados e quais as regras de encaminhamento de dados. A arquitetura é ilustrada da Figura 10.

A proposta de (SALAHUDDIN et al., 2014) é alterar a infraestrutura tradicional de VANET, com nuvens de RSUs. A arquitetura da nuvem consiste de RSUs tradicionais, *micro*

Figura 10 – Arquitetura RSU *Cloud*




**Fonte:** (SALAHUDDIN et al., 2014)

*datacenters* e máquinas virtuais que favorecem a programabilidade do SDN. Na nuvem, os fornecedores oferecem infraestrutura e aplicativos para serem alugados. A flexibilidade do SDN é usada para reconfigurar a rede, de acordo com o custo, para que os serviços virtuais possam ser migrados ou replicados atendendo as demandas e usando de forma eficiente os recursos da nuvem.

A técnica usada para esse gerenciamento é uma heurística que conta as mudanças nas regras de encaminhamento de dados, e nos números de serviços migrados. A exclusão e modificação de regras na tabela de fluxo inclui um custo que sobrecarrega o plano de controle. A solução é que tanto a tabela de fluxo, quanto as modificações das regras da tabela que estejam em um mesmo grupo sejam contadas duplamente, ter uma exclusão da regra antiga, e em seguida uma adição de uma nova regra. A sobrecarga que ocorre devido à migração da VM é simplificada para contar as mudanças nos *hosts* de serviço e não contabiliza a VM.

A contagem usada na heurística pode ser ampliada para ser uma medida mais complexa, sendo sensível a diferentes parâmetros, como tamanho das VMs, hora do dia. Com base nas análises de sobrecarga e reconfiguração, é então definida formalmente a reconfiguração da sobrecarga. A configuração da rede consiste em criar um *snapshot* contendo informações sobre os serviços dos *hosts* e os dados da tabela de encaminhamento, que consiste em fluxos e regras da tabela de grupo. As Nuvens de RSUs projetadas são eficientes quando se trata de gerenciamento de recursos na nuvem e minimizam um conjunto de migrações de VMs, sobrecarga do plano de controle, número de serviço *hosts* e atraso na infraestrutura.

Uma arquitetura de redes veiculares definida por software é proposta na pesquisa de(FONTES et al., 2017). O foco é a gestão de recursos em ambientes dinâmicos de veículos

e as potencialidades das redes veiculares definidas por software. Ainda é feita uma análise sobre a necessidade de mudanças na abordagem tradicional SDN dos pontos de vista teóricos e práticos. Avaliado por simulação de uma prova de conceito baseada na arquitetura desenvolvido no *Mininet-WiFi* para mostrar a aplicabilidade e alguns benefícios do SDN no cenário de uso proposto.

Em (FONTES et al., 2017), demonstra-se que é possível aplicar os princípios do SDN em VANETs, de forma prática e implementação real, que em muitos casos são ignoradas nas pesquisas. Porém, alguns problemas críticos são levantados ao tentar colocar SDN em VANET na prática, o que incluem entre eles; pilhas de *protocolos openflow*, implementação de extensão sem fio, escolha de controladores (*plugins* de protocolo, APIs), avaliação experimental, mobilidade e integração físico virtual.

A proposta de (FONTES et al., 2017) cria e implementa uma arquitetura de veículos para VANETs, nomeada de carros como nó (*node car*), adaptada para o emulador de rede Mininet-WiFi. A ferramenta tem suporte ao protocolo *OpenFlow* baseado em virtualização leve, que adiciona ao emulador a função de emular o canal sem fio e o suporte à mobilidade.

Uma prova de conceito foi elaborada pelos autores para experimentar os mecanismos de agregação de recursos e validar a arquitetura carro como nó, criada em um cenário VANET baseado em SDN. O controlador *OpenFlow* é responsável por gerenciar todos os nós emulados (carros, RSUs e Nós de estação base). O experimento realizado apresenta um carro que transmite vídeos durante um tempo determinado para uma central de operação para fins de segurança, controle de tráfego, e ou vigilância (são exemplos gravação de vídeo sobre acidentes, gravação de área para serem compartilhados com aplicações de nuvem). No caso emulado, o centro de operação pode estar localizado remotamente ou na borda da rede. No cenário demonstrado, o controlador *OpenFlow* seleciona quais tecnologias sem fio são mais adequadas no carro para transmitir o fluxo de vídeo.

A técnica abordada no trabalho de (FONTES et al., 2017) instala os fluxos estáticos e os modificam no *switch* de agregação com fio (cada *switches* possuem suas portas com diferentes pontos de acesso sem fio) para emular a mobilidade dos carros de forma controlada. Uma aplicação de vídeo *streaming* levada em consideração para os testes de casos. Durante a execução do experimento, o carro que envia o vídeo deixa a cobertura de uma estação base, então o controlado SDN detecta a nova condição e faz com que o carro se conecte a uma RSU ou a uma estação base para enviar o fluxo de vídeo ao vivo através das portas do *switch*.

O gerenciamento e desenvolvimento em redes sem fio usando SDN pode trazer inúmeras vantagens e desafios para VANETs . As pesquisas propõem várias técnicas, arquiteturas e novas funcionalidades para suportar os benefícios do uso de SDN em VANET. No entanto, apesar de toda flexibilidade, gerenciamento de recursos, melhoras no controle de desempenho sobre a heterogeneidade dos estados veiculares, o plano de controle centralizado ainda sustenta algumas ameças de seguranças e vulnerabilidades.

## 3.2 Virtualização de Funções de Redes em Redes veiculares

Toda a agilidade do serviço SDN é aprimorada pela virtualização de funções de rede NFV e sua capacidade de criar rapidamente, dimensionar ou realocar recursos virtuais. As funções são recursos transferíveis e controláveis (ONFTR-526, 2016).

Na literatura, a tecnologia de virtualização para ambientes VANETs foi usada no trabalho de(CHUNG; JIN, 2010) para virtualizar *gateway* com o intuito de prover segurança em redes veiculares. O papel fundamental do *gateway* é permitir a troca de mensagens entre redes internas e externas. A conexão externa permite uma exposição maior da rede, possibilitando que um comportamento do mundo externo inesperado ou mal-intencionado ocasione falhas no sistema. As falhas de sistema são uma das questões mais críticas a serem observadas nos veículos. Especialmente as falhas do sistema que ocorrem no *gateway* que podem ser propagadas para diversas áreas e causar sérios problemas para as operações veiculares sensíveis a ataques.

Os aplicativos dos veículos são baseados na colaboração entre nós em diferentes redes. Então a arquitetura de *gateway* de rede que tenha segurança pode evitar ou até mesmo isolar estas falhas que ainda são muito críticas. A proposta do artigo de (CHUNG; JIN, 2010) é criar uma rede veicular de *gateway* seguro, explorando os recursos da virtualização. Os recursos da virtualização foram usados para isolar falhas do sistema e impedir sua propagação para outros domínios. O aprimoramento da segurança pode ser definido como autenticação, criptografia, detecção de intrusão, filtragem. O esquema utiliza uma pilha de software para evitar a propagação das falhas para todo o sistema.

A ideia básica da proposta é ter dois sistemas operacionais no *gateway* de rede: um dedicado somente à rede externa e outro às redes internas. Para este trabalho a virtualização é usada para executar dois sistemas operacionais diferentes no *gateway* da rede. Assim, com esta divisão, os *gateways* das redes internas executam sistemas operacionais de tempo real que são verificados em termos de exatidão e segurança, para serem usados em redes internas sem modificações.

O *gateway* da rede externa monitora todo o comportamento inesperado das conexões externas, a proposta é considerar um sistema operacional que tenha sido aplicado em servidores de internet. Os sistemas operacionais convidados são executados em tempo real para redes externas e internas, respectivamente. O próximo passo é usar um monitor de máquina virtual (*Virtual Machine Monitor* - (VMM)), para isolar as falhas do sistema através de um sistema operacional convidado. O nível de segurança do encaminhamento de dados entre as redes pode variar de acordo com os dados que trafegam pelo nó do *gateway* (CHUNG; JIN, 2010).

## 3.3 Arquiteturas NFV e SDN para Redes Veiculares

Alguns estudos abordam diversas formas de melhorar a escalabilidade, gerenciamento flexível, dinâmico de redes veiculares. Nesta seção, serão descritos alguns trabalhos existente na literatura sobre esses temas.

O gerenciamento dinâmico de conexões veiculares é proposto em (CHEN et al., 2017). Segundo ele, uma arquitetura de redes veiculares definida por software (SDVN) é desenvolvida e capaz de resolver os desafios de fornecer conexões flexíveis e eficiente, garantia da qualidade de serviço e gerenciamento de múltiplas requisições. Adotando o paradigma de SDN uma abordagem de gerenciamento das conexões foi desenvolvida de forma centralizada.

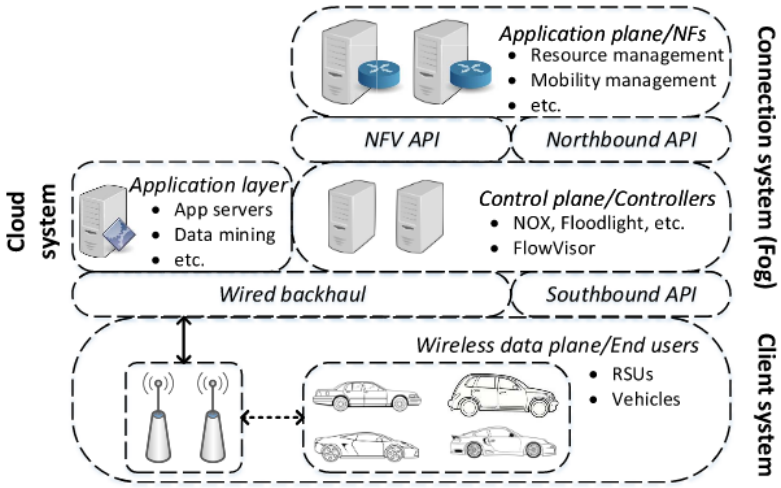
O trabalho mostra que o desempenho das comunicações entre os nós veiculares pode impactar em grande parte a qualidade dos serviços ITS. O protocolo de comunicação veicular, dedicado à comunicação de curto alcance não é suficiente para as demandas futuras de ITS baseada em carros conectados. Esta variedade de serviços requer que as conexões sejam flexíveis e eficientes entre veículos e RSU, bem como entre veículos. Além disso, os serviços oferecidos podem ter requisitos de QoS diferentes, que precisam ser cumpridos para melhorar a satisfação do usuário.

(CHEN et al., 2017) focam no gerenciamento dinâmico de conexões veiculares orientados para serviços usando SDN para cenário de redes veiculares. A solução tem como objetivo resolver os desafios ITS mencionados anteriormente, conexões flexíveis e eficientes, garantia de QoS dos serviços e suporte a múltiplos pedidos simultâneos. A arquitetura definida pelos autores de SD-IoV contribui com o desenvolvimento de processo de controle de conexões centralizadas, criação de rede de veículos com a função de garantir o QoS dos serviços, e a eficiência da utilização dos recursos, e a criação de dois algoritmos, um genético e um heurístico com menos carga computacional para resolver problemas de sobreposição da criação de rede de veículos (OVERLAY VEHICULAR NETWORK CREATION (OVNC)). A arquitetura SD-IoV é apresentada na figura 11.

Para a realização de experimentos, uma RSU tem a cobertura de um segmento de estrada. A RSU está conectada a um controlador SDN, permitindo que os veículos que estejam na área da cobertura, sejam controlados de forma centralizada. Os autores adotaram o paradigma da *fog computing*, então o controlador SDN está localizado na *fog*. A *fog* pode ser considerada em muito casos como uma nuvem leve, com menor latência na comunicação, o que é um desafio ainda crítico no cenário envolvendo SDN e VANET. Um plano de aplicação é criado para implementar as funções virtuais, gestão de recurso e gerenciamento de mobilidade.

Com os resultados obtidos e avaliação dos algoritmos,(CHEN et al., 2017) afirmam que a arquitetura proposta do *SD-IoV* exemplifica e facilita a integração das tecnologias de ponta, incluindo SDN, NFV e VANET, que integradas são responsáveis pela próxima geração de sistemas de transportes inteligentes. Além de mostrarem que os resultados do

Figura 11 – Arquitetura SD-IoV




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

gerenciamento da conexão veicular ou redes de sobreposição servem como padrão para a utilização de recursos em cenários SDN e VANETs.

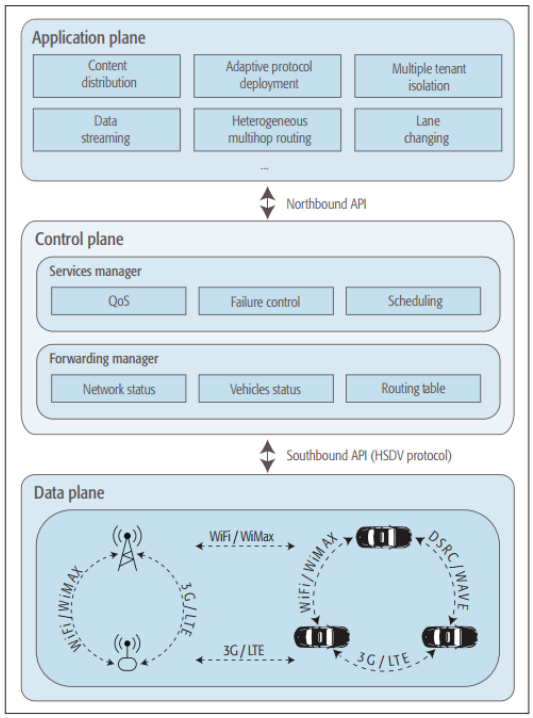
Uma arquitetura baseada em hierarquia SDN chamada HSDV para redes veiculares é proposta por (CORREIA; BOUKERCHE; MENEGUETTE, 2017) para solucionar problemas de desempenho em situações de perda de conexão com o controlador centralizado. Simulações são realizadas para mostrar cenários com o gerenciamento SDN em comparação com protocolos de roteamento tradicionais.

Um dos desafios quando ocorre a integração SDN/VANETs é tentar resolver a falta de conectividade entre os veículos e o controlador SDN centralizado. A condição de ter um único controlador para coordenar as rotas é uma desvantagem quando ele se torna inacessível. Na arquitetura tradicional sem SDN para resolver este problema, a solução é usar protocolos de roteamento tradicionais (CORREIA; BOUKERCHE; MENEGUETTE, 2017). A arquitetura proposta trabalha para resolver o problema de perda de conexão com o controlador, pode ser visualizada na Figura 12.

A solução apresentada foi usar domínios SDN locais através de *clustering* e uma hierarquia de controladores locais para acesso ao controlador principal. Quando não há conectividade com o controlador central os domínios SDN podem usar a inteligência da rede para recuperar as falhas de maneira mais eficiente.

Na arquitetura do HSDV o plano de dados é onde ocorre o gerenciamento das comunicações V2V, V2I com diferentes tecnologias de redes como WiFi e LTE. O plano de controle é responsável pelo gerenciamento dos tráfegos de dados e pela inteligência da rede, e possui dois módulos principais. Um módulo de gerenciamento de serviços para as requisições e para verificar quais os requerimentos de QoS necessários. O outro módulo é o de encaminhamento responsável pelo roteamento dos dados e análise geral da topologia da

Figura 12 – Arquitetura HSDV




**Fonte:** (CORREIA; BOUKERCHE; MENEGUETTE, 2017)

rede. Por fim, o plano de aplicação apresenta as aplicações veiculares e as interações com os sistemas baseados em SDN, são exemplos de aplicações vídeo *streaming*, distribuição de conteúdos entre outros.

Uma nova arquitetura para as redes veiculares definidas por software SDVN, foi proposta por (SUDHEERA et al., 2016), para dá prioridade aos requisitos de atraso em VANET. O trabalho apresenta os principais perfis de atraso e compara com outras arquiteturas SDVN.

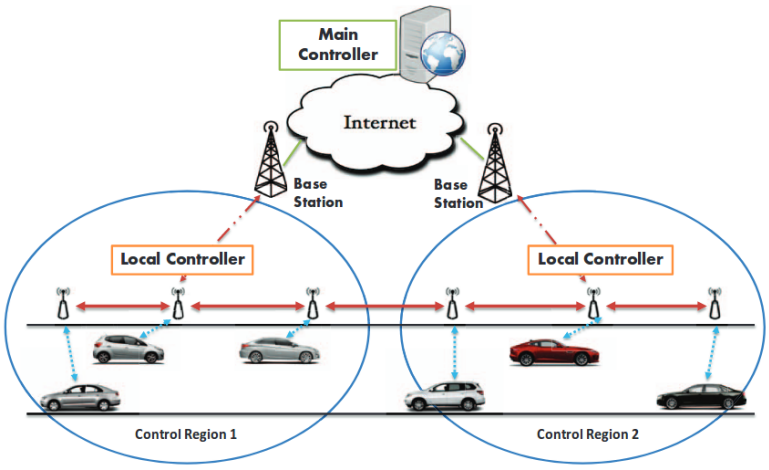
Na maioria das arquiteturas que utilizam SDN em VANETs, o controle da rede é feito através de um único controlador centralizado. Para este tipo de configuração, o controlador toma as melhores decisão baseado na visão global da rede. Geralmente, um *gateway* é fornecido para implantar aplicativos, uma vez que ele pode estar hospedado em servidor na internet. Portanto, como o controlador está longe dos veículos, a latência nas comunicações são significativas. A análise do atraso não é levada em consideração nas arquiteturas SDVN (SUDHEERA et al., 2016).

Como as redes veiculares são sensíveis ao atraso, uma das prioridades é dar atenção à latência gerada durante as operações em cada arquitetura. A solução de (SUDHEERA et al., 2016) trabalha com a nova arquitetura proposta para analisar de forma minuciosa os atrasos e demonstrar as vantagens com relação à outras arquiteturas existentes.

A proposta da arquitetura de redes veiculares definidas por software consiste de três requisitos básicos: uma configuração de rota, mas com um tempo de rota baixo, o uso

do DSRC como tecnologia sem fio para VANET, e ser capaz de fornecer flexibilidade e programabilidade à rede. A Figura 13 ilustra a arquitetura proposta por (SUDHEERA et al., 2016)

Figura 13 – Proposta de arquitetura SDVN




**Fonte:** (SUDHEERA et al., 2016)

De acordo com esta arquitetura, o plano de controle deve estar o mais próximo possível ao veículo para diminuir o atraso geral, de forma que os pacotes não passem pela internet para alcançar o controlador. Porém, se o controlador for retirado, perde-se a programabilidade e o *gateway* para acesso aos aplicativos. Para resolver este problema, (SUDHEERA et al., 2016) divide o plano de controle em dois níveis hierárquicos, colocando o controlador principal em nível superior na internet, e o segundo plano para o nível local das RSUs. Assim, as RSUs funcionam como controladores locais em uma determinada área, com uma visão global localizada.

## 3.4 Discussão

A arquitetura proposta nesta dissertação tem como contribuição prover gerenciamento flexível e dinâmico, com suporte à escalabilidade em redes veiculares, usando os paradigmas SDN e funções de redes virtuais. A proposta cria controladores SDN como funções de redes virtuais e de forma dinâmica, provendo flexibilidade em VANETs.

O trabalho de(SALAHUDDIN et al., 2014) também usa o SDN como um gerenciador de recursos para as redes veiculares. No entanto, esse tipo gerencialmente estático para VANETs ainda é um desafio. A nossa proposta diferencia da de (SALAHUDDIN et al., 2014) por virtualizar o plano de controle e replicá-lo, diferente de ter apenas um único controlador realizando todos os encaminhamentos e migração das máquinas virtuais com serviços. Essa diferença evita a sobrecarga que ocorre devido à migração de VMs e as reconfigurações.

O trabalho de (FONTES et al., 2017) também segue a linha de criar uma arquitetura para gerenciar recursos em ambientes de VANETs usando os princípios do SDN. Porém, não utiliza as vantagens que o NFV pode trazer aos ambientes SDVN. Diferente do que é proposto por esta dissertação, que procura oferecer as redes veiculares definidas por software uma infraestrutura que permite a dinâmica do plano de controle, resolvendo um dos problemas ao se ter um único controlador centralizado.

(CHUNG; JIN, 2010) buscam oferecer arquitetura de rede veicular com suporte à virtualização para flexibilidade e controle de segurança, mas a proposta exige que o ambiente VANET tenha um *gateway* de ligação com redes diferentes. Além disso, ao afirmar o uso de uma função de rede virtualizada (*gateway*), não fica claro que foi usada a arquitetura NFV, também não faz uso dos benefícios do SDN.

A integração das tecnologias SDN e NFV apresentada neste trabalho é mais uma contribuição que permite uma maior escalabilidade e controle da alta mobilidade do ambiente VANET. O trabalho de (CHEN et al., 2017) cria uma arquitetura usando SDN e NFV para melhorar a escalabilidade em VANETs através do gerenciamento dinâmico das conexões veiculares. No entanto, o gerenciamento foi criado de forma centralizada e estática, além da virtualização do plano de aplicação apenas transformar o servidor de aplicação em uma máquina virtual sem usar o padrão NFV ETSI.

Por último, (SUDHEERA et al., 2016) a propõe um gerenciamento levando em consideração os atrasos recorrentes em VANETs devido aos fatores do congestionamento. A arquitetura de (SUDHEERA et al., 2016), apesar de dar atenção à latência gerada durante as operações em SDVN, fornece a flexibilidade e programabilidade da rede. As RSUs funcionam como controladores locais, levando um alto custo de hardware, além de ter uma visão da rede limitada. O padrão NFV poderia resolver o problema de custo, mas não é utilizado. Em comparação com proposta desta dissertação, a hierarquia de controladores é usada para solucionar problemas nas perdas e atrasos da rede, no entanto os controladores são criados de forma estática.

A arquitetura proposta neste trabalho implementa o NFV de acordo com o padrão recomendado da arquitetura ETSI, o que para uso em ambientes de VANET pode ajudar a reduzir custos e aumentar a implantação e a velocidade de novos recursos. Uma outra vantagem em relação aos artigos já mencionados, é a capacidade de escalar os controladores, o que além de solucionar o congestionamento das redes veiculares, ainda é possível resolver o problema de controladores estáticos e centralizados. A Tabela 2 traz um resumo comparativo entre os trabalhos relacionados.

Tabela 2 – Tabela de Relacionados

| **Proposta** | **Objetivo** | **Problema** | **Tecnologias Usadas** | **Arquitetura SDN** | **Criação Dinâmica de Controladores** | **Desvantagens** |
|---|---|---|---|---|---|---|
| (CHUNG; JIN, 2010) | Virtualizar *gateway* com o intuito de prover segurança em redes veiculares. | Um comportamento do mundo externo inesperado ou mal-intencionado que ocasione falhas . | NFV | - | - | Não seguiu o padrão NFV, apenas usou o conceito de virtualização. |
| (SALAHUDDIN et al., 2014) | Criar uma arquitetura de nuvens de RSUs. | O Alto custo dos serviços de hospedagem em micro data centers. | SDN e Nuvem | Centralizada | - | Muitas modificações no plano de controle, provocando um atraso na rede. |
| (FONTES et al., 2017) | Criar um arquitetura SDN para gestão de recursos em ambientes dinâmicos veiculares. | Solucionar a falta de gerência nos recursos de comunicação e de rede no ambiente veicular. | SDN | Centralizada | - | controlador SDN centralizado. |
| (CHEN et al., 2017) | Gerenciamento dinâmico de conexões veiculares orientados para serviços usando SDN. | Falta de conexões flexíveis e eficientes. | SDN e NFV | Centralizada | - | NFV retratado no trabalho apenas como NFV API |
| Meneguette:2017 | Arquitetura para solucionar perda de conexão VANET com o controlador SDN. | Perda de conexão entre os veículos e o controlador SDN. | SDN | Hierarquia | - | Avaliação do algoritmo apenas no controlador primário. |
| (SUDHEERA et al., 2016) | Uma arquitetura VANET/SDN para dar prioridade aos requisitos de atraso em VANET. | Problemas de latência em redes veiculares definidas por software. | SDN | Hierarquia | - | O número de controladores nas RSUs tem um custo de processamento. |
| Solução Proposta | Uma arquitetura flexível, dinâmica e escalável para VANET/SDN baseada em NFV. | Falta de gerenciamento flexível e dinâmico com suporte à escalabilidade. | SDN e NFV | **Dinâmica** | ✓ | Cenário Veicular limitado, tempo de criação de novos controladores. |

## 3.5 Considerações Finais

Este capítulo apresentou uma discussão sobre os trabalhos relacionados à criação de novas arquiteturas envolvendo SDN e NFV para gerenciar e solucionar problemas da falta flexibilidade, suporte ao caráter dinâmico e escalabilidade em redes veiculares, os vários trabalhos mostraram a relevância desta pesquisa. O uso do SDN mostra ganhos significativos no gerenciamento de recursos, a arquitetura com criação dinâmica de controladores SDN resulta na programabilidade e flexibilidade da rede.

Nesta dissertação, o NFV é utilizado para virtualizar as funções dos controladores, reduzir custos e permitir a execução de várias serviços VNFs em infraestrutura construída por meio da computação em nuvem, enquanto outros trabalhos usam apenas o SDN como controlador central para escolhas de redes, ou o NFV apenas como uma virtualização simples, sem o uso de todos os seus componentes.

# 4 PROPOSTA

Este capítulo tem por finalidade descrever a proposta da arquitetura para prover gerenciamento flexível, dinâmico e com suporte à escalabilidade em redes veiculares definidas por software usando NFV. A Seção 4.1 apresenta os serviços da arquitetura, e a Seção 4.2, o ambiente de teste utilizado.

## 4.1 Descrição da Proposta

A fim de prover o suporte ao gerenciamento flexível, dinâmico e com suporte à escalabilidade em redes veiculares, este trabalho propõe uma arquitetura SDVN-vC, *Software defined vehicular network with virtualized controller*. A proposta considera, inicialmente, elementos pertencentes ao plano de controle e plano de dados. No plano de dados os elementos consistem na infraestrutura básica de redes veiculares, composta por RSUs e veículos. No plano de controle, a proposta integra os componentes de serviços do *Openstack*, com os padrões NFV e virtualiza a função do controlador SDN.

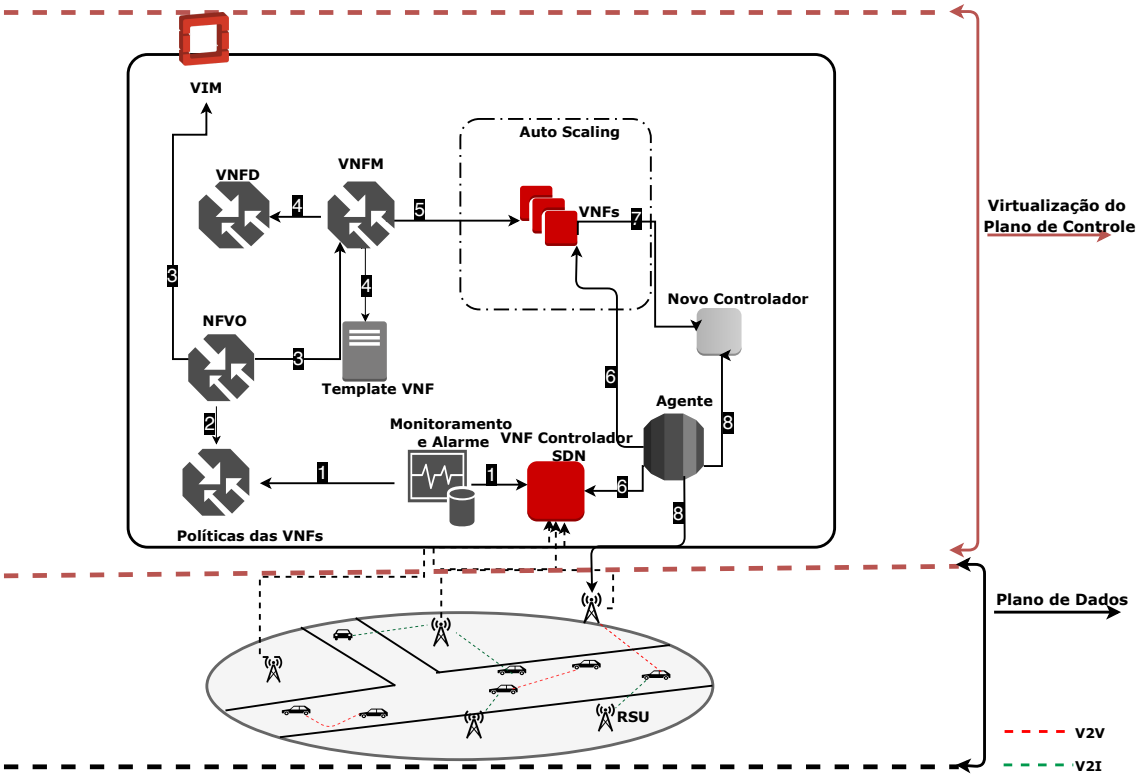
A arquitetura foi concebida e elaborada com base na ordem de eventos no processo de execução dos serviços. Os componentes estruturais incluem organização total e visão geral da comunicação; formas de acesso aos dados, consultas, processos de alarmes, monitoramento e políticas das VNFs, além dos tipos de comunicações veiculares adotados.

Com o intuito de atingir os objetivos da proposta, é necessário que o desenvolvimento siga alguns critérios. Os seguintes requisitos foram definidas para a criação da arquitetura:

1. Gerenciar a camada de virtualização.
2. Monitorar as VNFs.
3. Definir políticas de redimensionamentos.
4. Criar VNFs de forma automática e dinâmica.
5. Comunicação veicular com controladores remotos.

Em um ambiente real, a nuvem *openstack* foi configurada com seus principais serviços: autenticação, armazenamento, rede, computação e orquestração. O NFV baseado na arquitetura *ETSI MANO* foi integrado à nuvem por meio do serviço *tacker*, um projeto do próprio *openstack* para gerenciamento e orquestração de VNFs e serviços de redes usando a plataforma NFV. O *tacker* possui uma *API* responsável pelo ciclo de gerenciamento das VNFs, com capacidade de monitorar a VNF, realizar auto escalonamento e autorecuperação.

Figura 14 – Arquitetura SDVN-cV para VANETs




**Fonte:** Elaborada pelo Autora

O *Openstack* é usado como gerenciador de infraestrutura virtual, de acordo com os padrões NFV chamado de VIM. Para definir os serviços de redes (NETWORK SERVICE DESCRIPTORS (NSD)) e descrever as VNFs (VIRTUALIZED NETWORK FUNCTION DESCRIPTORS (VNFD)s), o *tacker* usa uma topologia e especificação de orquestração para serviços de nuvem, conhecida como TOSCA e ela é responsável por descrever as topologias das aplicações para a rede e para o ambiente de nuvem que incluem todos os componentes. As configurações, modificações e criação de VNFs são realizadas usando *templates*. Os controladores SDN são definidos com estes *templates*, onde CPU, memória, disco, rede, segurança, e as políticas de escalonamento são configuradas.

Para uma visão de redes veiculares definidas por software, o controlador SDN na nuvem pertence ao plano de controle para gerenciar e orquestrar as comunicações veiculares. O plano de dados é composto das comunicações entre veículos e RSUs. A proposta da arquitetura virtualiza o plano de controle para uma maior flexibilidade e como estratégia

que pode ser adotada para prover auto *scaling*. Na arquitetura, o orquestrador NFVO é responsável por consultar os recursos que o VIM possui e quais políticas de VNFs foram definidas, que assegurem uma instanciação eficiente de acordo com o número mínimo e máximo de máquinas a serem criadas, quando os alarmes do monitoramento forem disparados.

O VNFM, que gerencia as VNFs, consulta o VNFD onde foram criados e armazenados os discos virtuais dos *templates*, e quais as políticas foram atribuídas. Então, através dos templates TOSCA, o *auto scaling* é realizado. Então, o resultado é um novo controlador SDN para dividir a carga com o controlador sobrecarregado. E um agente monitora todo o processo de *Scaling Out* e notifica às RSUs.

O controlador SDN criado como VNF, possui uma regra de encaminhamento de pacotes para portas conhecidas, com o objetivo de gerenciar a carga e a troca de informações na rede. Quando uma nova requisição chega ao controlador, um evento *Packet In* ocorre e o controlador grava as informações de porta de origem e destino no *switch*, para que de uma próxima vez não seja mais necessária uma nova requisição ao controlador. Na programação SDN, esse tipo de processo é conhecido como *learning switch*. No plano de controle, o controlador gerencia todas as trocas de mensagens que ocorrem nas comunicações veiculares V2V e V2I.

Uma das maneiras de verificar se um controlador SDN está congestionando, é analisar as perdas de pacotes na rede e o tempo de resposta das requisições. Ele pode suportar milhares de novas requisições. Porém esse número crescente pode fazer com que a rede sofra alterações no fornecimento de serviços. Desta forma, a arquitetura contempla uma estratégia baseada na carga de trabalho do controlador com intuito de tornar o ambiente escalável, realizando operações de alocação de forma automática.

## 4.2 Implementação da proposta

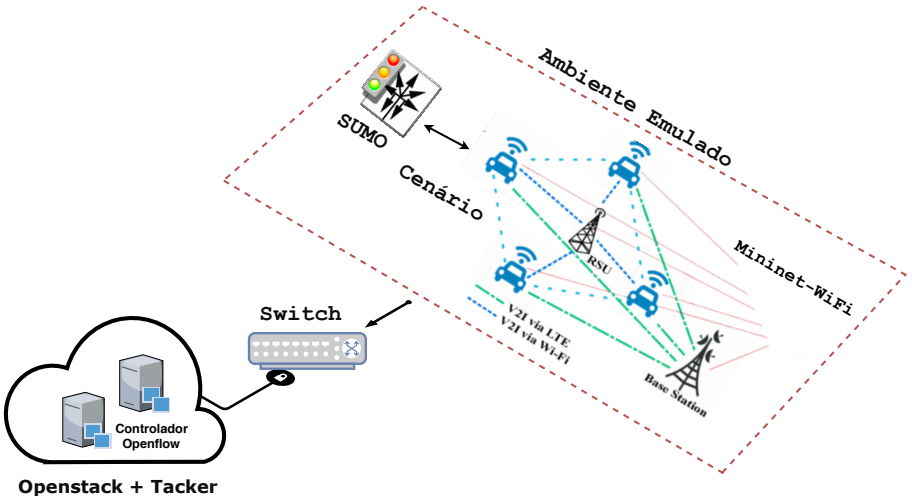
O ambiente de testes da arquitetura com suporte à escalabilidade, onde os experimentos foram realizados, será discutido nesta seção. Todo o ambiente de testes foi montando em um laboratório de pesquisa localizado no Centro de Informática, campus da UFPE (Universidade Federal de Pernambuco).

Para a realização dos experimentos, uma nuvem privada foi configurada com o *openstack* e a arquitetura NFV foi integrada através do orquestrador *Tacker*. Com esse ambiente, é possível definir políticas de escalonamento baseadas nas condições de uma função de rede virtual. Para isso, são usados os serviços de monitoramento*ceilometer* e alarme *aodh* do *openstack*.

Foi definido como limiar no arquivo de políticas que a condição para iniciar um novo controlador ocorreria quando a CPU atingisse 60%. Um template pré-configurado usando a linguagem yaml é utilizado pra criar uma VNF com uma quantidade de memória, CPU

e disco definidos, além das configurações de rede. A Figura 15 ilustra a visão física do ambiente de teste.

Figura 15 – Ambiente Físico




**Fonte:** Elaborada pelo Autora

Um servidor *DELL* executa o sistema operacional Linux, distribuição ubuntu 16.04, com o emulador Mininet-WiFi instalado. O ambiente de nuvem foi configurado em um servidor IBM. Um cenário VANET foi criado para caraterizar a mobilidade dos carros em conjunto com o SUMO, responsável por injetar tráfego diretamente nos controladores. Todas as requisições e carga de trabalho são feitas com as trocas de mensagens e solicitações da topologia criada no emulador *Mininet wifi*. A Tabela 3 de configuração do *hardware* é apresentada a seguir:

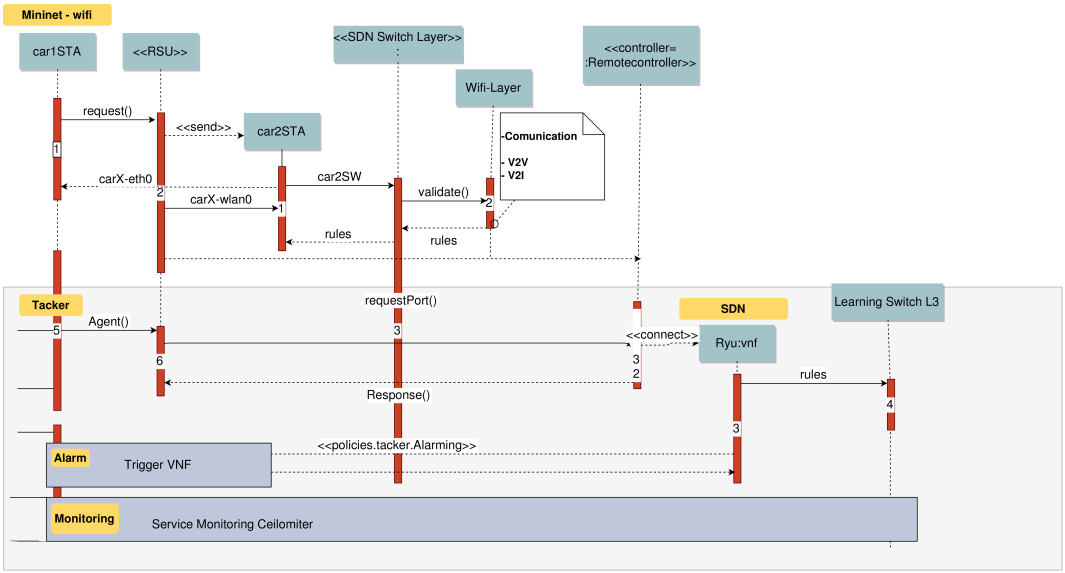
| **Características** | **Configuração de Hardware** | **Configuração** |
|---|---|---|
| Nuvem e NFV | Servidor Processador Intel (R)Xeon(R) CPU X3363, Quad-core 2.83GHz, Memória RAM de 16GB, Tipo de VNF: flavor-c1.small, 1VCPU, e 512 MB de memória | Ubuntu Server 16.04 Com controlador Ryu - Nuvem Openstack |
| Mininet-WiFi | Servidor Processador Intel (R)Core(TM) i7 Quad-core 2.10GHz, Memória RAM de 12GB, | Ubuntu Server 16.04 *Mininet-WiFI* |

Tabela 3 – Configuração Física

Um agente na nuvem é usado para enviar a notificação de mudanças de controladores para as RSUs, quando surgem as novas réplicas de controladores, para dividir a carga de requisições e mensagens entre elas e o controlador. A divisão é realizada da seguinte forma: as RSU1 e RSU2 permanecem no controlador principal, e as RSU3 e RSU4 são configuradas para usar o são configuradas para usar o controlador 2, o que ocorre em tempo

de execução. A sinalização da comunicação e os passos do funcionamento da proposta são apresentados na Figura 16

Figura 16 – Diagrama de Sinalização




**Fonte:** Elaborada pelo Autora

Na Figura 16 é mostrado um cenário mais detalhado do que acontece na comunicação veicular integrada com o controlador na nuvem, orquestrado pelo *tacker*. O cenário é descrito da seguinte forma:

1. Inicialmente, ocorrem trocas de mensagens entre os veículos (V2V). Os carros possuem interfaces para as conexões com a camada intermediária onde, *CarX, carXsw e carXSTA* representam a entidade carro como nó. Nessa primeira etapa, todos os fluxos entre os carros e entre carros e RSUs são identificados pelo controlador SDN.

2. As interfaces *carXwlan0* são usadas para as comunicações V2I. Para que o tráfego ocorra entre os carros com diferentes interfaces, é necessário realizar um roteamento estático para não lidar com pacotes *arp*. A quantidade de interfaces para cada nó carro é uma exigência para gerenciar a comunicação V2V, uma vez que são conectadas diretamente. Por meio da camada Wi-Fi, a RSU é conectada ao controlador e realiza as interações com os carros.

3. Usando o emulador *Mininet-WiFi*, a conexão entre os dois ambientes é feita diretamente. Através da Camada SDN usando a função *RetroC ontroller* e o método *addController*, um controlador localizado na nuvem é adicionado à rede. A execução ocorre na porta 6653, de onde chegam as requisições da rede veicular. Neste processo o controlador recebe muitos pacotes *PacketIn*, que pode ser observado em determinados momentos da simulação.

④ As regras SDN definidas para este cenário são adicionadas utilizando uma aplicação padrão no controlador Ryu, onde são identificados os endereços de origem e destino e portas de acesso.

⑤ Ocorre o monitoramento das VNFs usando a API *policies.tacker.Alarming*, onde uma política é descrita no *template* TOSCA para definir um alarme com a ação de *scaling*. Através do serviço VNF manager, o *tacker* gerencia as VNFs usando o serviço de alarme do *openstack* chamado *aodh*, para enviar notificações ao serviço de telemetria *ceilometer*. Com os dados armazenados no *ceilometer*, o *tacker* NFV usando suas políticas definidas e baseadas nos lineares do *template*, cria uma nova função de rede virtual, um novo *Ryu:vnf*.

⑥ Um agente é usado na nuvem para gerar notificação às RSUs à medida que novos controladores SDN venham a surgir na rede, para realizar a divisão de carga. O agente checa se algum serviço na porta 6653 está executando em alguma VNF a cada 1 segundo. Ao perceber que já está disponível um outro controlador para prover a escalabilidade, o agente envia uma notificação para todas as RSUs, para adicionar um novo controlador em seus sistemas. No cenário da proposta, as RSUs são conhecidas e os IPs dos controladores foram definidos nos *templates* de modo estáticos.

### 4.2.1 Agente de Notificação - Pseudocódigo

Um agente é utilizado para checar a existência de um novo controlador e para relizar o balanceamento entre as VMs na mudança de controladores. Após este passo, uma mensagem é enviada para cada uma das RSUs. O pseudocódigo a seguir representa o funcionamento do agente.

**Algoritmo 1:** Agente de Notificação

```
Data: IP do Host
Result: Change Controller
Starting;
while host True in controller do
    for host in controller;
    socket ← 6653;
    if True then
        RSU ← IPs Address ;
        agent.connect;
        set-controller;
    else
        return False There is no new controller ;
    end
end
```

O agente foi implementado em *python*, tornando a comunicação com o controlador e a nuvem mais simples, visto que, tanto o *Openstack*, quanto o controlador *Ryu* são plataformas desenvolvidas nesta linguagem. O algoritmo **1** descreve o funcionamento do agente, a função básica do programa é verificar a cada 1 segundo se existe um novo controlador SDN em execução na nuvem. Uma lista com os possíveis endereços do controlador configurados nos *templates* do *TOSCA* é armazenada, e então, através de uma função *socket*, é testada uma conexão na porta 6653 que a padrão do controlador.

Caso essa condição seja verdadeira, o agente enviará uma alerta para as primeiras RSUs, com a mensagem de mudança de controlador. Caso a condição da conexão seja falsa, o programa entende que não há novos controladores e volta a realizar os testes novamente. Todo o processo é realizado em tempo de execução.

Para o bom funcionamento do agente, é necessário conhecer os endereços dos controladores SDN que já foram configurados de forma estáticas nos *templates* dasVNFs, bem como o endereço e a placa das RSUs que estão sendo executadas no *Mininet-WiFi* para possibilitar o chaveamento das RSUs com os novos controladores.

# 5 AVALIAÇÃO E RESULTADOS

Nesta seção, são apresentados os experimentos realizados para avaliar o comportamento do cenário de redes veiculares definidos por software, usando apenas um controlador central como VNF e em situações de auto escalonamento. Para avaliar a necessidade de criar uma nova função virtual, apenas a variável de utilização da CPU foi levada em consideração. No entanto, no futuro, outras variáveis, como o uso de memória RAM, podem ser consideradas na definição da política. As condições da rede são avaliadas através da métrica de perda de pacotes durante o tempo de simulação. O capítulo está organizado da seguinte forma: A Seção 5.1 apresenta o ambiente de teste; A Seção 5.2 apresenta parâmetros e métricas utilizados. Os resultados da prova de conceito com e sem a arquitetura flexível e dinâmica proposta, e com o uso de uma aplicação de transferência de arquivo são discutidos na Seção 5.3.

## 5.1 Ambiente de Teste

Foram criados três cenários veiculares diferentes diversificando a quantidade de carros. O objetivo foi usar o cenário da comunicação e da troca de mensagens veiculares para congestionar controladores SDN. A mobilidade é definida pelo arquivo de configuração do SUMO com as posições dos carros de acordo com o mapa da cidade de Nova York. Os carros fazem seus trajetos conforme o mapa estabelecido. Para cada movimentação do veículo, a posição desse veículo é atualizada no Mininet-WiFi. O tempo de simulação definido para cada execução foi de 60 segundos, período em que foram identificadas as trocas de dados entre os carros e as unidades de acostamentos, além de requisições de dados.

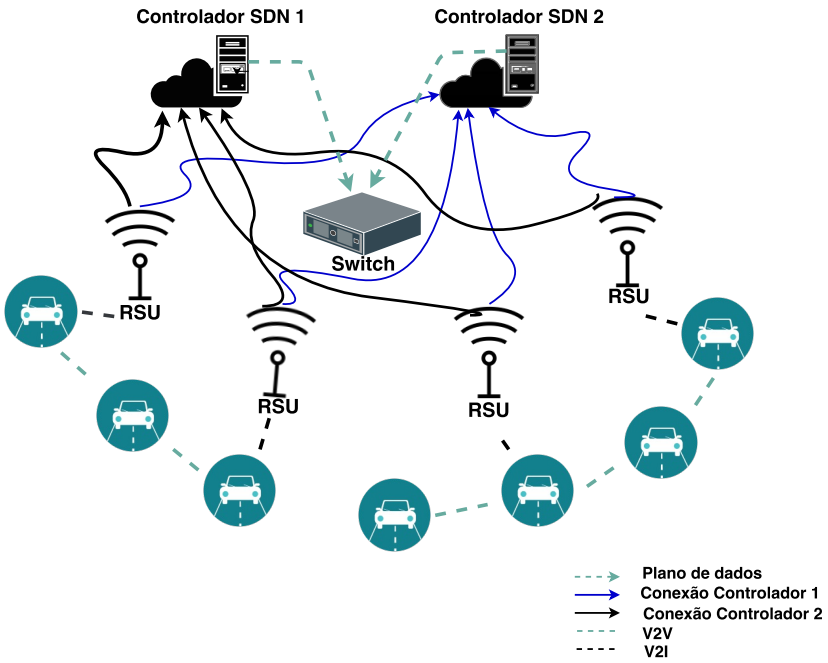
Cada RSU está conectada a um controlador SDN remoto e nesse ambiente elas funcionam como *Accesses Points*. Com a mobilidade dos carros, o tráfego é gerado no código do cliente "nó", também chamado *car0*, que envia requisições a um servidor de arquivo, localizado na nuvem. Para avaliar qual o impacto de possíveis congestionamentos em redes veiculares, uma simulação usando o gerador de tráfego (*Iperf*) foi utilizada para mostrar requisições de aplicações de troca de arquivos através da comunicação V2I. A Tabela 4 apresenta os principais parâmetros utilizados na configuração.

| | |
|---|---|
| Tempo da Simulação | 60s |
| Veículos | (120) (260) (310) |
| Velocidade Máxima | 55,56 m/s |
| Taxa de Transferência | 20Mbps |
| RSUs | 4 |

Tabela 4 – Configuração da Emulação

A Figura 17 ilustra o cenário lógico produzido no Mininet-WiFi com RSUs conectadas aos controladores, todas as RSUs criadas são conectadas a todos os controladores existentes e os novos adicionados. Para efeitos de distribuição de cargas nas situações de congestionamento, as conexões das RSUs serão divididas sempre que possível pela metade da quantidade existente. Tanto as comunicações V2V, V2I e as conexões com os controladores remotos são configuradas no cenário.

Figura 17 – Representação Lógica do cenário veicular no Mininet-Wifi




**Fonte:** Elaborada pelo Autora

A avaliação da proposta foi conduzida em um *tesbed* de nuvem privada integrado a um emulador de redes veiculares e então foram realizadas comparações entre a condição de usar uma rede veicular definida por software convencional e com a proposta de SDVN-cV. A avaliação dos quatro experimentos, sendo o primeiro sem o uso da escalabilidade e em seguida experimentos demonstrando o redimensionamento usando dois, três e quatro controladores que, por sua vez, teve como objetivo avaliar a solução proposta neste trabalho.

Para definir qual a porcentagem da CPU a ser usada nas políticas de *auto scaling* ,

um teste de limiar foi realizado para monitorar a VNF durante o tempo da simulação. É possível identificar que no início da simulação, no instante no qual ocorre um aumento no processamento representado pelo valor de 60% atingido pela CPU, começam a ocorrer perdas de pacotes. Os gráficos preliminares mostram o período de monitoramento e o momento das perdas de pacotes capturadas pela ferramenta *wireshark*.

Figura 18 – Análise da CPU durante a simulação

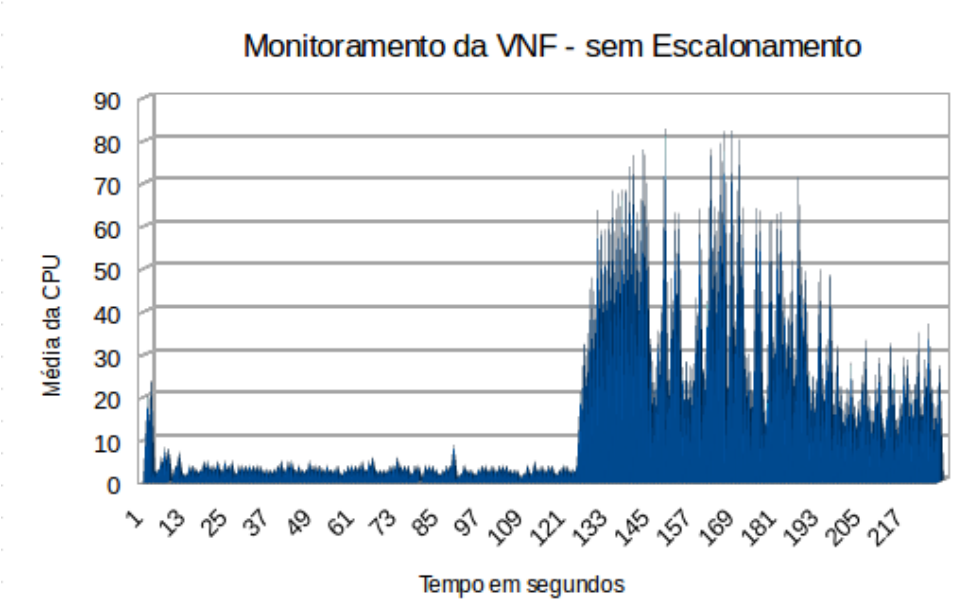
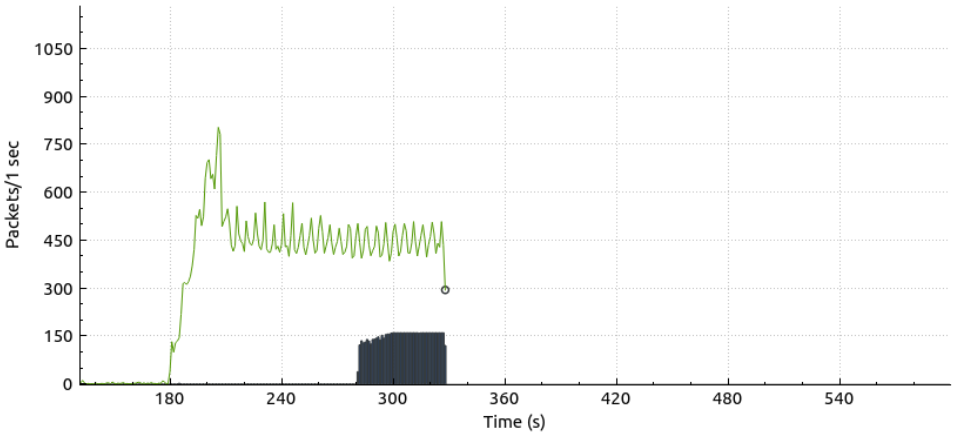



Figura 19 – Análise da perda de pacotes




Para os testes iniciais, com o objetivo de definir os limiares de medições das métricas, o processamento da VNF *controller* é monitorado. A topologia de redes veiculares foi iniciada com todas as mensagens de trocas entre o cenário elaborado e o controlador, com o intuito de mostrar o impacto da comunicação no processamento da CPU. A sincronização do tempo com o *wireshark* não é precisa, visto que a ferramenta deve ser iniciada alguns segundos antes para selecionar as interfaces a serem monitoradas. Quando ocorre o aumento do uso da CPU, percebe-se uma perda de pacotes na rede devido ao alto processamento de novas requisições.

A Figura 18, apresenta uma amostra da média da CPU monitorada antes, durante e depois do início do experimento. Nos segundos iniciais, a VNF está com poucos processos e ainda sem requisições. A partir do instante 130s, a VNF *controller* recebe as mensagens packet-in e mensagens dos carros. Nesse período, a média da CPU atinge mais de 60% e em determinados momentos chega aos 80%.

A perda de pacotes na Figura 19, capturados através da ferramenta wireshark, são representadas por pequenas barras azul, as linhas verdes são as quantidade de pacotes transmitidos por segundos durante toda a comunicação. O tempo das perdas não são simultâneos com o tempo da CPU da VNF por tratar-se de duas formas de coleta de dados. Usando os filtros de coletas, percebe-se que na placa de rede isolada para as transmissões, nos segundos iniciais, a linha verde representa todos os pacotes transmitidos. A ferramenta foi inicializada alguns segundos antes dos cenários do Mininet-WiFi com o SUMO. Aproximadamente entre os segundos 280 e 330 a perda de pacotes é referente às mensagens perdidas durante o tempo da simulação veicular.

## 5.2 Métricas e Parâmetros

Para avaliar o desempenho da aplicação veicular, utilizo-se o *jitter* e a perda de pacotes durante o período do experimento. Toda a injeção de tráfego é caracteriza pelo cenário VANET simulado. Os fatores e parâmetros são definidos na emulação, para alterações na quantidade de carros que são acrescentados para cada tipo de experimento, assim como o aumento de números de RSUs. Estas alterações são necessárias para gerar mais cargas e simular o congestionamento dos controladores SDN escalonados.

Uma das estratégias usadas para avaliar o congestionamento de uma máquina é medir sua porcentagem de uso da CPU. Esta métrica foi considerada para tomadas de decisão de redimensionamento da VNF. A variação da CPU é bastante sensível à grandes quantidades de requisições e troca de dados e, portanto, é bastante usada em computação em nuvem para avaliação do estado das VMs. Além disso, a sobrecarga da máquina faz com que menos pacotes sejam processados, acarretando perdas de pacotes e atrasos.

Tabela 5 – Projeto dos Experimentos

| **Características** | **Descrição** |
|---|---|
| Sistema | Ambiente de Computação em Nuvem com *OpenStack* e arquitetura *Tacker*, emulador *Mininet-WiFi*. |
| Métricas | *Jitter* e perda de pacotes das aplicações, medição do processamento da CPU, perda de pacotes na rede veicular. |
| Parâmetros | CPU, quantidade de VNFs |
| Fatores | Configuração do *Mininet-WiFi* (quantidade de carros, número de RSUs, tempo da simulação). |
| Carga de Trabalho | Carga de trabalho dos veículos no *Mininet-WiFi*. |
| Projeto de experimentos | O experimento consiste da geração da carga de trabalho caracterizado por um ambiente de redes veiculares para medições de perda de pacotes, *Jitter* e CPU.O **Experimento 1** foi executado uma aplicação de transferência de arquivos apenas com uma VNF controlador. **O Experimento 2** aplicação executada em um ambiente de *auto scaling* com duas VNFs. No **Experimento 3** foram três VNFs e por fim o **experimento 4** foi realizado com quatro VNFs |
| Análise dos Dados | Interpretação dos resultados descritos nos gráficos. |
| Apresentação dos Resultados | Estatística Descritiva, Intervalo de Confiança e Gráficos . |

## 5.3 Resultados e Discussões

Os experimentos usando a arquitetura VSDN sem escalabilidade e os demais testes com os acréscimos de VNFs, como controladores SDN, serão apresentados e discutidos.

### 5.3.1 Arquitetura sem a proposta com apenas um controlador

Na nuvem, apenas um controlador SDN é iniciado como VNF para a topologia de redes veiculares elaborada no Mininet-WiFi. A quantidade de carros e as RSUs são usadas para simular um ambiente de congestionamento. Neste cenário, um dos carros requisita dados de um servidor de arquivo. Com estes elementos, podemos afirmar que temos um conjunto de fatores para realizar uma carga de trabalho no controlador e medir o comportamento de sua CPU. Toda a mobilidade dos veículos na rede é responsabilidade do SUMO. Este teste foi repetido 30 vezes para fins de garantir a confiabilidade estatística.

A Figura 20 ilustra uma das coletas com o desempenho da CPU durante o tempo da simulação. A imagem mostra os picos de variação da CPU da VNF durante a simulação VANET quando apenas um controlador está sendo executado, para atender solicitações

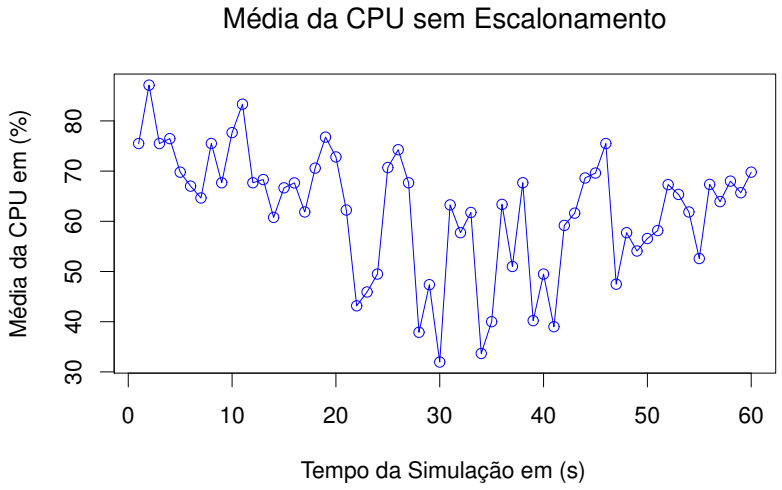



Figura 20 – Uso da CPU

dos veículos. Como as primeiras requisições são de *Packet-in*, e requerem um certo processamento do controlador, a CPU tem uma maior porcentagem de uso. No decorrer da execução, as regras do controlador fazem com que apenas novas mensagens passem por ele, pois as demais já foram aprendidas e armazenadas no *switch*. Durante esse período, novos carros podem passar pela RSU conectada ao controlador e mais uma vez ocorre a elevação na carga de processamento da CPU.

Neste experimento é possível perceber que a VNF controlador não tem um aumento da média da CPU constante durante o tempo de simulação. Quando o número de mensagens *packet-in* enviada do switch para o controlador diminuem, a CPU reduz seu processamento, apresentado nos instantes que a média fica entre 30% e 40%. A média geral dos testes mostra a CPU com 64,15% com intervalos entre 56,54 e 71,76% para uma confiança de 95%. As demais estatísticas descritivas, como mediana e desvio padrão, serão discutidas, comparadas e detalhadas com as abordagens escaláveis nas próximas subseções.

### 5.3.2 Arquitetura SDVN-cV

Para esta subseção, serão discutidos os resultados obtidos ao realizar um estudo de caso da arquitetura com suporte à escalabilidade, implementando redimensionamentos com dois, três e quatro controladores SDN como VNFs.

#### *5.3.2.1 SDVN-cV com dois controladores*

Nesta primeira avaliação, um controlador está em funcionamento para atender as solicitações veiculares, ao atingir a média de CPU delimitada no orquestrador NFV, apenas uma VNF será criada, o valor do incremento para a criação de instâncias está configurado para (1). Com a ação de *scaling out*, agora existem dois controladores para suprir as

demandas das comunicações na rede. Os dados da Figura 21 são referentes às execuções dos dois controladores.

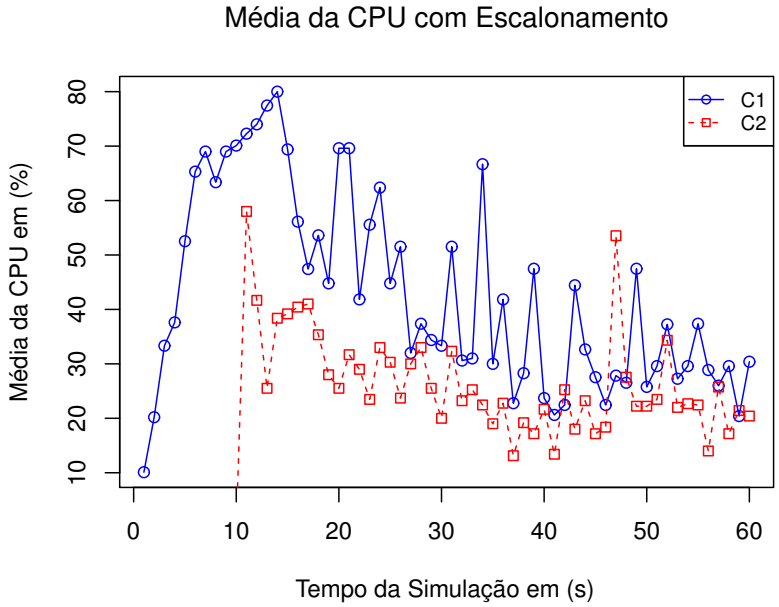



Figura 21 – Gráfico do Uso da CPU com dois controladores

Para a coleta final, os dados da CPU capturados durante o tempo de iniciação da VNF e do controlador são descartados, visto que poderiam afetar os resultados reais do experimento. O gráfico 21 exibe média da CPU durante os 60 segundos da simulação, ao atingir 60% do uso de CPU da primeira VNF representada pela legenda C1, mesmo que controlador 1, uma nova VNF é criada, chamada de C2 na legenda do gráfico ou controlador 2. As VNFs são criadas com os endereços de redes, armazenamento e serviços pré-configurados no template.

Ao iniciar o novo controlador, a carga das RSUs será dividida entre eles. Esta divisão é feita baseada no número de RSUs criadas, por *default* a metade delas proporcionam um equilíbrio entre os controladores. Mesmo que em um determinado momento ocorra um maior fluxo de carros em uma RSU, e uma outra esteja menos congestionada, a política da divisão em 50% continua, a escalabilidade é para solucionar os problemas de fornecimento de serviços pelos controladores e não a escalabilidade das RSUs. No gráfico da figura 21 uma das amostras retrata o momento que uma nova VNF é acionada durante o experimento.

O comportamento das CPUs analisado durante a injeção de tráfego VANETs mostram, que nos primeiros segundos, o primeiro controlador ou C1 conforme apresenta a legenda do gráfico 21 está com alto processamento. Este evento dura mais de 15 segundos, onde uma nova instância é iniciada para amenizar a sobrecarga. Quando os dois controladores passam a trabalhar juntos como réplicas, a porcentagem de uso é reduzida e se mantêm em torno de 30% a 40% . Para evitar uma nova sobrecarga e consequentemente mais um

processo de *scaling*, o agente na nuvem envia mensagens de configurações para que as RSUs dividam suas requisições.

O percentual de uso do segundo controlador representado no gráfico como C2, é exibido depois de quase 10 segundos, isso ocorre porque até esse momento o controlador SDN *RYU* não havia sido instanciado. Apesar desse tempo de instanciação, o limiar usado de 60% provê uma margem considerável para estas operações. Até o final da simulação ainda ocorrem aumentos na média da CPU, mas não chega a ser constante como nos primeiros segundos, a nova VNF instanciada não atinge 60% da média em nenhum período. A Tabela 6 resume as estatísticas nas abordagens escalonadas e sem escalonamento.

| **Uso da CPU** | | | | |
|---|---|---|---|---|
| Abordagem | Média | Mediana | Desvio | Intervalo de Confiança |
| SDVN-cV Controlador 1 | 37,21 | 31,47 | 16,13 | $25{,}66 \leq x \leq 48{,}75$ |
| SDVN-cV Controlador 2 | 35,58 | 29,53 | 16,62 | $23.69 \leq x \leq 47.47$ |
| Sem a proposta | 64,15 | 67,01 | 10,64 | $56{,}54 \leq x \leq 71{,}76$ |

Tabela 6 – Estatísticas Descritivas para dois controladores

A Tabela 6 apresenta as estatísticas descritivas para os dados de uso da CPU durante todos os testes. Pode-se perceber que a média de processamento com apenas uma VNF é praticamente duas vezes maior em relação ao ambiente com escalabilidade. Outra métrica importante é o desvio padrão que mostra o espalhamento dos dados em torno da média, onde outra vez a escalabilidade é mais vantajosa, em média 16,13% e 16,62% contra 10,64%, com e sem a proposta, respectivamente. A Tabela 2 mostra também a média geral dos experimentos, mediana (50% dos dados) e o intervalo com 95% de confiança.

#### *5.3.2.2 SDVN-cV com três controladores*

Para este experimento, a intenção foi avaliar a proposta usando três controladores, agora existem dois controladores que são congestionados para que ação de criar uma nova VNF aconteça. Os parâmetros no Mininet-WiFi também são modificados, porque são necessários mais carros e troca de informações para enviar o maior número de mensagens *Packet-In* possíveis. O número de 260 veículos e 4 RSUs foram suficientes para que os dois controladores atingissem uma média superior a 60%. A figura 22 mostra a carga média de processamento na CPU das VNFs durante o tempo da simulação em situações com três controladores.

A partir da análise do gráfico, conclui-se que apesar de atingir o limiar de 60% nos primeiros segundos da simulação, a ação de *scaling* no Tacker NFV não é efetuada. A execução da tarefa ocorre apenas próximo aos 20 segundos. Com relação ao escalonamento mostrado na subseção anterior, as VNFs passam mais tempo congestionadas, ou seja, o

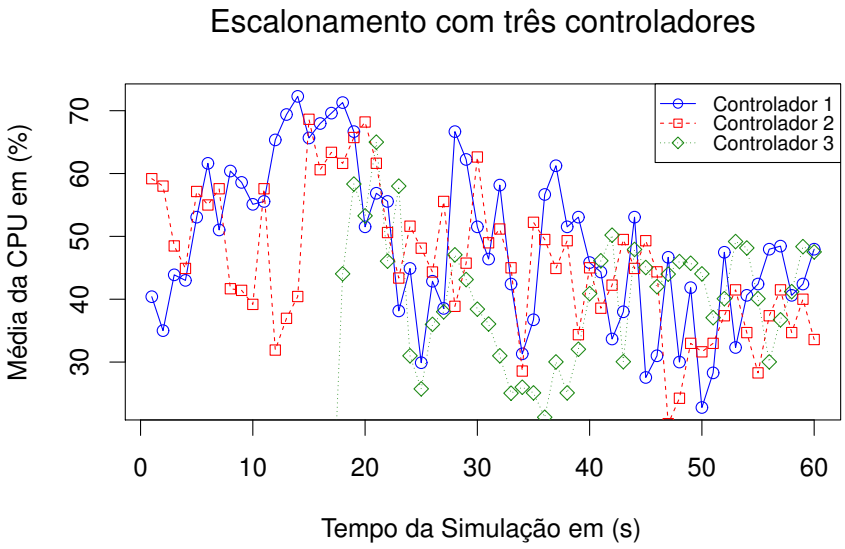



Figura 22 – Gráfico do Uso da CPU de três controladores

tempo para acionar uma nova instancia é maior. A explicação é o uso do *ceilometer* nos monitoramentos, que agora tem mais uma VM para coletar dados e também exige mais recursos da nuvem.

O terceiro controlador é criado para executar em paralelo com os dois já existente. E todo o processo para separação das RSUs ocorre novamente. Logo após o segundo 20, o controlador 3 atinge 60%, mas logo em seguida cumpre sua função de estabilizar a média da CPU. Observa-se ainda que as médias de processamento das VNFs com relação ao experimento com dois controladores foi superior. A Tabela 7 mostra as estatísticas para o escalonamento com os três controladores.

| **Processamento da CPU** | | | | |
|---|---|---|---|---|
| Abordagem | Média | Mediana | Desvio | Intervalo de Confiança |
| SDVN-cV Controlador 1 | 40,49 | 37,25 | 15,87 | $29{,}14 \leq x \leq 51{,}85$ |
| SDVN-cV Controlador 2 | 38,04 | 32,27 | 17,77 | $25{,}32 \leq x \leq 50{,}75$ |
| SDVN-cV Controlador 3 | 43,76 | 44,01 | 16,717 | $31{,}80 \leq x \leq 55{,}72$ |
| Sem a proposta | 64,15 | 67,01 | 10,64 | $56{,}54 \leq x \leq 71{,}76$ |

Tabela 7 – Escalonamento com três controladores

As colunas da tabela contém os dados referentes as 30 execuções para cada tipo de abordagem. A abordagem sem escalonamento está sempre presente nas tabelas para enfatizar e comparar os resultados. O grau de variação entre as amostras das propostas mostram através do desvio padrão, e da tendência central que apesar dos casos em que a média chega a atingir mais de 60%, as médias gerais da arquitetura proposta SDVN-cV

não ultrapassam esse valor. Além disso, a variabilidade apresentada pelos valores pode ser justificada pelo fato de que em determinados momentos, faz-se necessário maiores cargas de trabalhos para ocasionar o aumento proposital do uso da CPU.

#### 5.3.2.3 SDVN-cV com quatro controladores

Para este quarto experimento, o objetivo é fundamentar os casos de necessidades de uma maior instanciação dos controladores SDN como VNFs. O intuito é reproduzir o comportamento de uma determinada cidade com um número maior de carros, justificado, por exemplo, em grandes eventos: como a copa, ou olimpíadas. Com este aumento de fluxo veicular, e inúmeras quantidades de trocas de mensagens, mais controladores precisam ser escalonados.

A geração da carga pelo cenário VANET agora foi configurada com 310 carros e o número de RSUs continua 4 RSUs. Como já listados nas subseções anteriores sobre as demandas para congestionar o controlador glssdn, os parâmetros precisam ser alterados em cada experimento. O comportamento da utilização média da CPU sob a demanda de requisições, é apresentado no gráfico da figura 23. As linhas do gráfico representam os momentos de picos durante o tempo da simulação. Os controladores VNFs são representados na legenda como C1 para o primeiro controlador, C2 para o segundo, C3 para o terceiro e o quarto é representado por C4.

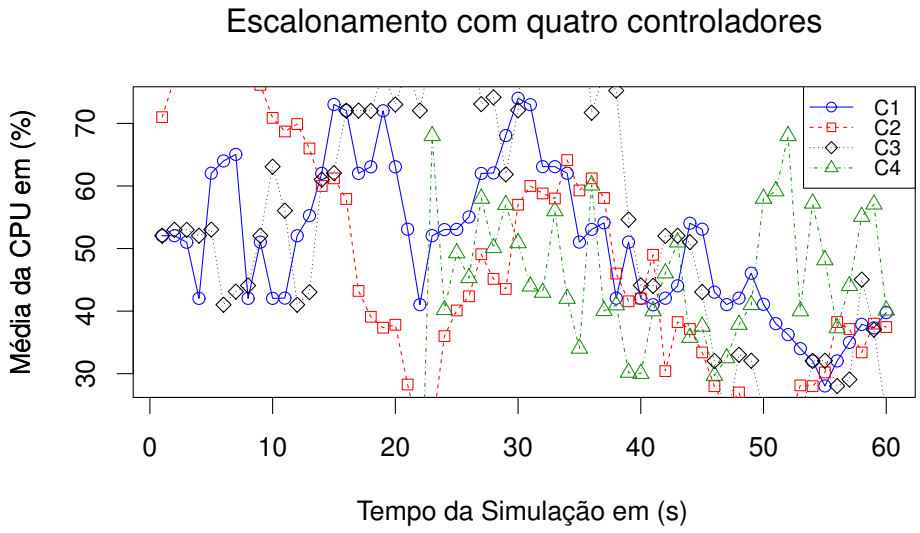



Figura 23 – Gráfico do Uso da CPU de quatro controladores

A exibição do gráfico permite constatar que para este experimento, apenas os três controladores VNFs estão sendo executados antes dos 30 segundos. Na configuração da operação para criar novas tarefas de *scaling*, as três VNFs tem suas CPUs monitoradas. Quando qualquer uma das máquinas virtuais atingir 60% uma nova é criada. Este

comportamento foi configurado nos templates TOSCA obedecendo o máximo, mínimo e incremento das VNFs. Ao observar as oscilações, nota-se que em nenhum momento os três controladores ficam congestionados ao mesmo tempo, ou seja, quando o C1 e C3 atingem picos de aproximadamente 70% de uso da CPU, pois ainda existe tráfego veicular durante todo o tempo da simulação, o C2 está com o precessamento baixo.

Continuando a análise, mesmo o controlador 1 atingindo 60% de uso da CPU antes dos demais, a nova VNF tem um atraso na criação, onde ocorre próximo aos 23 segundos da simulação. Depois de instanciar o novo controlador, ainda ocorrem picos de congestionamentos nos segundos iniciais, mas, em nenhum momento é possível elevar os processamentos de todos os controladores . A média de processamento da CPU do controlador C4 ainda atinge mais de 60%, o que se conclui que criar apenas um controlador não foi suficiente para amenizar o cenário. No entanto, com a limitação da nuvem, foi delimitado apenas o incremento de uma VNF a cada ação de *scaling out*. O intervalo de confiança para as médias, a mediana e o desvio padrão das 30 amostras serão apresentados na Tabela 8.

| **Processamento da CPU** | | | | |
|---|---|---|---|---|
| Abordagem | Média | Mediana | Desvio | Intervalo de Confiança |
| SDVN-cV Controlador 1 | 48,28 | 47,14 | 13,45 | $38{,}65 \leq x \leq 57{,}90$ |
| SDVN-cV Controlador 2 | 51,78 | 52 | 15,43 | $40{,}74 \leq x \leq 62{,}82$ |
| SDVN-cV Controlador 3 | 58,72 | 57,595 | 16,55 | $46{,}88 \leq x \leq 70{,}56$ |
| SDVN-cV Controlador 4 | 50,93 | 51,04 | 14,13 | $40{,}82 \leq x \leq 61{,}04$ |
| Sem a proposta | 64,15 | 67,01 | 10,64 | $56{,}54 \leq x \leq 71{,}76$ |

Tabela 8 – SDVN-cV com quatro controladores

A média das amostras aumenta de acordo com o número de controladores, o que pode ser constatado na tabela. Significa que quando existem mais VNFs controladores, suas médias de consumo de CPU foram maiores. Isso deve-se ao fato de um aumento maior na carga de trabalho, caracterizadas pelos números de veículos, RSUs e mensagens *Packet-In*. A partir destas análises e a comparação com a abordagem sem escalonamento, são notáveis os benefícios no desempenho, e na otimização do uso de processamento.

Além desses testes de monitoramento da CPU, também foram medidas as perdas de pacotes durante os experimentos, nos quais foram possível identificar quanto são essas perdas de pacotes, em ambientes de redes veiculares congestionadas, e qual o seu impacto em aplicações de troca de dados. A Subseção 5.3.2.4 traz maiores detalhes sobre as medições desta métrica.

#### 5.3.2.4 Perda de Pacotes durante os Experimentos

Ao executar os experimentos aumentando a quantidade de controladores para os escalonamentos, a perda de pacotes foi medida durante esses testes. Para cada execução, usando dois controladores, três, quatro e sem escalonamento. Nesta subseção serão apresentados os resultados para cada abordagem.

A Figura 24 apresenta a média e o intervalo dados com 95% de confiança, o gráfico retrata as perdas de pacotes na rede durante o período do aumento da CPU nas VNFs. Sem escalonamento, ou seja, usando apenas um controlador, a perda de pacotes é superior comparada com todos os cenários escalonados. Com o alto processamento da CPU, a VNF controlador processa as novas requisições, porém outras são retransmitidas ocasionado a perda de pacotes.

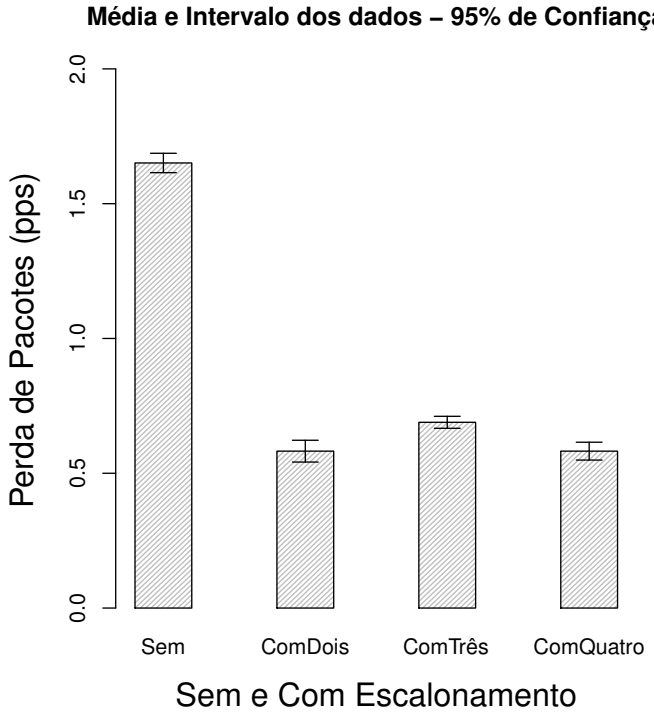



Figura 24 – Média da Perda de Pacotes por segundo

Durante o aumento da CPU, foi avaliada a perda de pacotes da simulação, de forma que quando havia apenas um controlador as medições de entrada e saída de dados da comunicação foram feitas com base na retransmissão de dados. Com a solução de auto *scaling* programado, ao dividir a carga com um outro controlador é possível notar que a perda de pacotes são menores nos dois controladores.

Na proposta de escalonamento também pode-se perceber que a perda de pacotes ocorre no início de eventos referentes à chegada de vários pacotes *packet-in*, porém nos próximos segundos ocorre uma leve redução. Como uma réplica da VNF está em execução e agora

processa menos pacotes, pois não recebe todo o tráfego VANET, os fatores de mobilidade dos carros, quantidade e velocidade continuam os mesmos.

A perda de pacotes apresentada nos controladores escalonados são semelhantes, ficando entre 0,5 e 0,7 pacotes por segundos. A justificativa para a redução é que agora existem mais VNFs controladores executando com maior possibilidade de desempenho. Assim, ao processar as várias mensagens *Packet-In* originadas dos cenários veiculares, mesmo com a alterações dos parâmetros no Mininet-WiFi, as retransmissões são reduzidas. A Tabela 9 descreve as estatística para a perda de pacotes em cada experimento realizado.

| Perda de Pacotes | | | | |
|---|---|---|---|---|
| Abordagem | Média | Mediana | Desvio | Intervalo de Confiança |
| Com Dois Controladores | 0,58 | 0,56 | 0,251 | $0{,}42 \leq x \leq 0{,}74$ |
| Três Controladores | 0,69 | 0,72 | 0,254 | $0{,}51 \leq x \leq 0{,}87$ |
| Quatro Controladores | 0,58 | 0,61 | 0,23 | $0{,}41 \leq x \leq 0{,}75$ |
| Sem Escalonamento | 1,65 | 1,57 | 0,53 | $1{,}27 \leq x \leq 2{,}03$ |

Tabela 9 – Estatística Descritiva para perda de pacotes

De acordo com os dados representados na tabela 8, a média da perda de pacotes da proposta escalável mostra-se favorável em relação a sem proposta. Quando dois controladores são usados, a redução média é 2,84 vezes menor. Ao usar três controladores escalonados, o número é 2,39 vezes. Com quatro controladores, levando em conta o intervalo de confiança pode-se afirmar que as perdas de pacotes são praticamente iguais. Porém, os valores podem ser maiores se o tráfego perdurar por mais tempo, pois ocorreriam mais requisições e mais veículos.

## 5.4 Avaliação de uma aplicação de transferência de arquivo

### 5.4.1 Medindo a perda de pacotes

Para identificar qual o impacto quando se tem perda de pacotes e um alto processamento, um experimento para avaliar o impacto na QoS de uma aplicação foi realizado. Para este fim, a aplicação foi caracterizada utilizando-se a ferramenta *Iperf,* sendo possível realizar requisições de dados em Mbps. A taxa de *download* foi ajustada para 20Mbps durante a simulação. Desta forma, foi possível simular o *download* ou uma transferência de informações entre dois pontos. Estes dois pontos são representados pela requisição do carro simulado ao servidor de arquivo na nuvem.

Comparando e analisando os resultados, pode-se observar que ocorre perda de pacotes e aumento de CPU no ambiente VANET com apenas um controlador. Com a solução proposta para prover o escalonamento, ocorre uma redução na perda de pacotes e no uso de CPU. Porém, para uma avaliação mais realista, uma aplicação de transferência de arquivo foi executada durante as simulações para descobrir qual o impacto dessas perdas e congestionamento em aplicações VANETs. Na Figura 25 é possível verificar a média e o intervalo dos dados da perda de pacotes durante as simulações sem e com a proposta usando dois controladores.

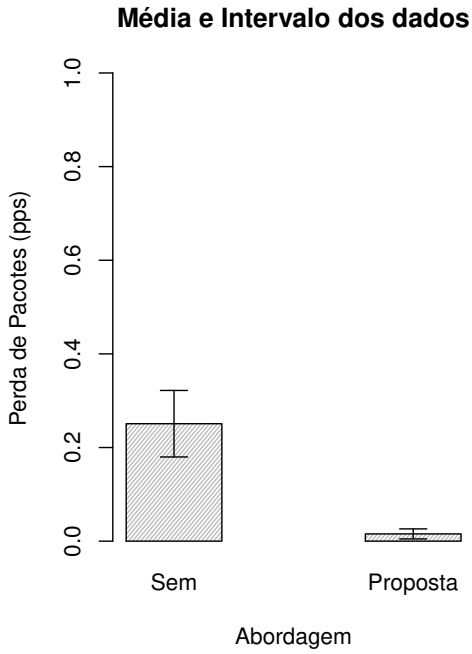



Figura 25 – Perda de Pacotes Aplicação - 95% de confiança

A Figura 25 exibe a comparação das perdas entre os controladores VNFs nas abordagens. Observa-se que a média das amostras representa 0,25 pacotes por segundos (pps) quando a aplicação foi executada no ambiente sem a proposta. E nas condições onde ocorreram o escalonamento, esta perda ficou com média de 0,015 pps. Ou seja, foi possível reduzir em 16,6 vezes a perda de pacotes durante a simulação veicular. Portanto, usando a arquitetura apresentada com a solução de controladores instanciados dinamicamente de acordo com a estratégia proposta, é possível reduzir esta perda de maneira significativa.

| Perda de Pacotes | | | | |
|---|---|---|---|---|
| Abordagem | Média | Mediana | Desvio | Intervalo de Confiança |
| Com escalabilidade | 0,015 | 0 | 0,038 | $0 \leq x \leq 0{,}038$ |
| Sem Escalonamento | 0,250 | 0,19 | 0,256 | $0{,}096 \leq x \leq 0{,}405$ |

Tabela 10 – Estatística Descritiva para perda de pacotes - Aplicação

A Tabela 10 resume as estatística resultantes da coleta de dados referente apenas à perda de pacotes da aplicação veicular. Quando executados nos cenários com escalabilidade, pode-se afirmar que dentro do intervalo de confiança não se teve uma perda de pacotes expressiva. Levando em consideração o intervalo de confiança de 95% para a média das amostras, ao relizar sem escalonamento, a perda de pacotes chega a ser de 16,6 vezes superior.

### 5.4.2 Medindo o Jitter

O objetivo de medir o jitter da aplicação, deve-se ao fato de obter os resultados da qualidade da rede. Assim, no caso dessa métrica que mede a variação da latência, quanto menor for a variação, melhor será o desempenho da rede e melhor será a qualidade do veículo cliente.

Em comparação com uma proposta escalável, é visível o ganho com relação ao jitter sem escalabilidade. A variação do atraso na entrega de dados na rede na condição sem *scaling*, mostra-se superior ao da condição com *scaling*. No gráfico da figura 26 são apresentados a média e o intervalo dos dados da aplicação, com um resumo da coleta de dados de 30 execuções. O impacto da proposta com dois controladores pode ser observado na figura 26.

A Figura 26 mostra a concentração de dados quando a aplicação foi executada nos dois cenários, primeiro sem escalonamento e depois com o uso de escalonamento dinâmico dos controladores SDN. Quando a medição é realizada durante os experimentos com escalonamento e em determinadas amostras não ocorrem atrasos o jitter é 0. No momento em que os testes foram realizados sem escalonamento, a média e o intervalo representados no gráfico são superiores, mostrando que sempre existe variância no atraso dos pacotes.

| **Jitter na Aplicação** | | | | |
|---|---|---|---|---|
| Abordagem | Média | Mediana | Desvio | Intervalo de Confiança |
| Com escalabilidade | 0,165 | 0,177 | 0,050 | $0{,}134 \leq x \leq 0{,}196$ |
| Sem Escalonamento | 0,239 | 0,213 | 0,091 | $0{,}184 \leq x \leq 0{,}294$ |

Tabela 11 – Estatística Descritiva do Jitter na Aplicação

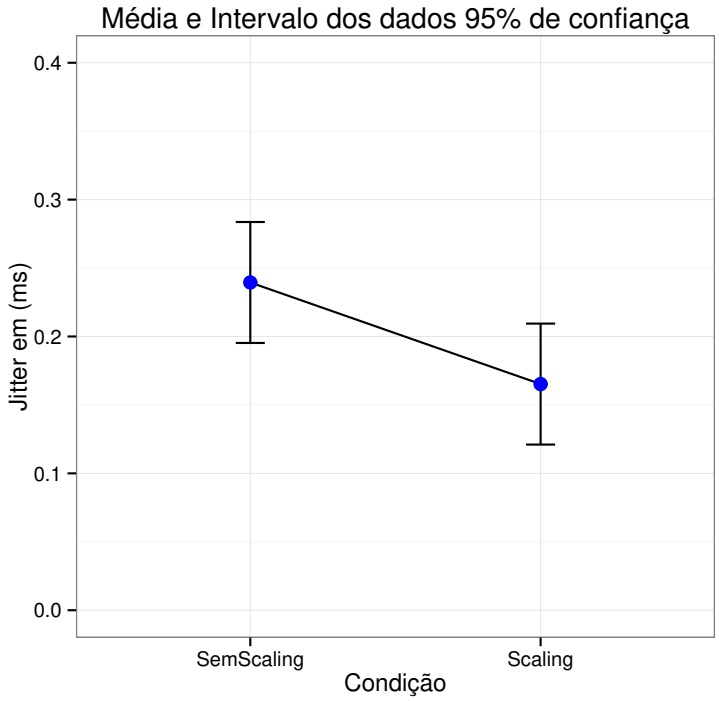



Figura 26 – Gráfico do Jitter em (ms )

A Tabela 11 resume a representação dos dados com médias e intervalo de confiança das medições. Podemos afirmar que em 95 de 100 amostras hipotéticas para o jitter, o resultado estará dentro deste intervalo. Baseado nessa hipótese realizados, ou seja, o risco relativo estará entre 0,134 e 0,196 para abordagem com escalabilidade. Para a arquitetura sem escalabilidade os dados estarão entre 0,184 e 0,294. De acordo com os dados apresentados na Tabela 11, o jitter teve suas médias de tempo de atraso reduzidas de 0,239 ms para 0,165 ms, uma diferença de redução de 43,75%.

Ao realizar esses testes com esta aplicação de troca de arquivos, onde no cenário VANET um veículo realiza uma requisição para nuvem, através da comunicação V2I, percebe-se que, de acordo com os resultados apresentados, é correto afirmar que existe um impacto em aplicações veiculares. Ou seja, quando a rede veicular sofre interferência ou no congestionamento ou no processamento, isso pode impactar no fornecimento de serviços. Assim, a arquitetura aqui proposta pode ser usada para beneficiar aplicações sensíveis a atrasos em suas comunicações.

## 5.5 Considerações Finais

Este capítulo apresentou e analisou os dados coletados durante os teste de redes veiculares usando uma arquitetura com suporte à escalabilidade e sem escalabilidade, e comparou os resultados mostrando ganho em nível de aplicação ao usar a proposta do trabalho.

# 6 CONCLUSÃO

Este capítulo resume os principais pontos discutidos neste trabalho, as considerações finais desta dissertação são apresentadas na Seção 6.1. Na Seção 6.2, são descritas algumas contribuições obtidas e, em seguida, na Seção 6.3, serão discutidos alguns pontos que podem ser explorados em trabalhos futuros.

## 6.1 Considerações Finais

Este trabalho apresentou uma arquitetura para o gerenciamento de redes veiculares definidas por software, usando NFV, com o intuito de gerenciamento flexível, dinâmico e com suporte à escalabilidade. Discutiu-se a necessidade de novas tecnologias para as redes veiculares, foram introduzidos novos conceitos como: redes definidas por software, virtualização de funções de rede, programabilidade, gerenciamento em VANETs, controladores SDN. Foram discutidos orquestradores NFV, *OpenMANO*, o NFV, *tacker* que foi utilizado para implementar toda a arquitetura NFV. Em seguida, foram apresentadas as redes veiculares, a arquitetura do *tacker*, abordando todos seus serviços de monitoramento e escalonamento, até sua implementação no ambiente *Openstack*.

Nesta dissertação, também foram discutidos a implementação de controladores SDN como funções de redes, a motivação para virtualizar as funções do plano de controle e os benefícios de gerenciamento, escalabilidade, flexibilidade e baixo custo da união entre as tecnologias SDN e NFV. A avaliação da arquitetura sem escalonamento e da proposta escalável foi realizada utilizando-se tanto métricas de redes quanto de QoS para aplicações sensíveis a perdas. Os resultados mostraram que a proposta possui um bom desempenho, e que estratégias que permitam o suporte à escalabilidade são indispensáveis para aliviar o congestionamento das redes veiculares e a qualidade de serviços de aplicações para os usuários.

## 6.2 Contribuições

Esta seção apresenta as principais contribuições obtidas nesta dissertação.

1. **Criação da Arquitetura SDN/NFV para VANETs** A principal contribuição foi gerenciar controladores SDN como funções de redes virtuais. Este é o primeiro trabalho, que se tem conhecimento, a utilizar integração das duas tecnologias para fins de gerenciamento flexível, dinâmico e com suporte à escalabilidade em redes veiculares.

2. **Implementação do padrão NFV** A segunda contribuição deste trabalho foi a implementação de toda a arquitetura NFV com seus componentes e de acordo com os padrões definidos pelo ETSI. E implementar a integração com o ambiente de computação em nuvem para se beneficiar de seus serviços de automação.

3. **Desenvolvimento de Agente de monitoramento**

   Outra contribuição relevante deste trabalho foi a criação de um agente simples e funcional para realizar o chaveamento das RSUs com os controladores virtuais, evitando a necessidade de usar um *Proxy*, ou um balanceador de cargas, o que ocasionaria um consumo maior nos recursos da nuvem.

## 6.3 Trabalhos Futuros

Esta seção apresenta e discute possíveis extensões à pesquisa desenvolvida nesta dissertação.

- **Monitoramento Inteligente** - O monitoramento das VNFs foi realizado através dos serviços de telemetria do *Openstack* , que tem como limitação coletar apenas algumas métricas de nuvem relacionadas à máquina virtual. A ideia é criar um monitor inteligente capaz de coletar várias métricas de uma VNF, como perda de pacotes, tempo de respostas, quantidade de requisições e baseado nelas acionar alarmes para ações de escalonamento;

- **Realização de experimentos com cenários mais complexos** - Usar ambientes emulados ou *testbed* que possibilitem uma maior quantidade de recursos. Tanto para os experimentos usando o Mininet-WiFi, quanto os realizados na nuvem usando cenário reais, a necessidade de recursos foi uma das dificuldades do trabalho;

- **Experimentos com aplicações de tempo real e de *streamig* de vídeo** - Neste trabalho foi usada uma simulação com o *iperf* para solicitar *downloads* de dados entre dois nós finais, uma nó carro e um servidor em nuvem. No futuro, pretende-se usar vídeos em tempo real e com uso de provedores de conteúdos.

- **Agente de Notificação Inteligente** -

  O agente na nuvem utilizado para enviar notificações às RSUs precisa ter sua base de dados configurada de forma estática. Futuramente, criar um agente capaz de identificar novos endereços na rede de forma dinâmica e configurar e remover controladores SDN nas RSUs para alocar e liberar recursos.

# REFERÊNCIAS

AL-DHURAIBI, Y.; PARAISO, F.; DJARALLAH, N.; MERLE, P. Elasticity in cloud computing: State of the art and research challenges. *IEEE Transactions on Services Computing*, v. 11, n. 2, p. 430–447, March 2018. ISSN 1939-1374.

AL-SULTAN aif; AL-DOORI, M. M.; AL-BAYATTI, A. H.; ZEDAN, H. A comprehensive survey on vehicular ad hoc network. *Journal of Network and Computer Applications*, p. 204–217, February 2013.

ALVES, R.; CAMPBELL, I. do V.; COUTO, R. de S.; CAMPISTA, M. E. M.; MORAES, I. M.; RUBINSTEIN LUíS HENRIQUE M. K. COSTA, O. C. M. B. D. M. A. M. G. Minicurso: Cap. 5 - redes veiculares: Princípios, aplicações e desafios. In: *SBRC-2009 Simposio Brasileiro de Redes e Sistemas Distribuidos*. [s.n.], 2009. Disponível em: <http://www.academia.edu/11638705/MINICURSO_Cap._5-Redes_Veiculares_Princ%C3%ADpios_Aplica%C3%A7%C3%B5es_e_Desafios>.

BATRA, N.; SINGH, R. P. A hybrid approach to increase the scalability in vanets. In: . [S.l.: s.n.], 2017.

BHOI, S. K.; KHILAR, P. M. Vehicular communication: a survey. *IET Networks*, v. 3, n. 3, p. 204–217, September 2014. ISSN 2047-4954.

BORCOCI, E.; AMBARUS, T.; VOCHIN, M. Distributed Control Plane Optimization in SDN-Fog VANET. *The International Symposium on Advances in Software Defined Networking and Network Functions Virtualization*, n. c, p. 124–130, 2017. Disponível em: <https://www.iaria.org/conferences2017/SOFTNETWORKING.html>.

BUYYA, R.; YEO, C. S.; VENUGOPAL, S.; BROBERG, J.; BRANDIC, I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, v. 25, n. 6, p. 599–616, 2009. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X08001957>.

CHEN, J.; ZHOU, H.; ZHANG, N.; XU, W.; YU, Q.; GUI, L.; SHEN, X. Service-oriented dynamic connection management for software-defined internet of vehicles. *IEEE Transactions on Intelligent Transportation Systems*, PP, n. 99, p. 1–12, 2017. ISSN 1524-9050.

CHUNG, S. M.; JIN, H. W. Isolating system faults on vehicular network gateways using virtualization. In: *2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*. [S.l.: s.n.], 2010. p. 791–796.

CORREIA, S.; BOUKERCHE, A.; MENEGUETTE, R. I. An architecture for hierarchical software-defined vehicular networks. *IEEE Communications Magazine*, v. 55, n. 7, p. 80–86, 2017. ISSN 0163-6804.

DLR. *SUMO – Simulation of Urban MObility*. 2017. Disponível em: <dlr.de/ts/sumo>.

ETSI, N. *Network Functions Virtualization - Architectural Framework*. 2013. Disponível em: <http://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf>.

FONTES, R. D. R.; CAMPOLO, C.; ROTHENBERG, C. E.; MOLINARO, A. From theory to experimental evaluation: Resource management in software-defined vehicular networks. *IEEE Access*, v. 5, p. 3069–3076, 2017. ISSN 2169-3536.

FONTES, R. R.; AFZAL, S.; BRITO, S. H. B.; SANTOS, M. A. S.; ROTHENBERG, C. E. Mininet-wifi: Emulating software-defined wireless networks. In: *2015 11th International Conference on Network and Service Management (CNSM)*. [S.l.: s.n.], 2015. p. 384–389.

GALANTE, G.; BONA, L. C. E. de. A survey on cloud computing elasticity. In: *2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*. [S.l.: s.n.], 2012. p. 263–270.

HAN, B.; GOPALAKRISHNAN, V.; JI, L.; LEE, S. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, v. 53, n. 2, p. 90–97, Feb 2015. ISSN 0163-6804.

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

HERRERA, J. d. J. G.; VEGA, J. F. B. Network functions virtualization: A survey. *IEEE Latin America Transactions*, v. 14, n. 2, p. 983–997, Feb 2016. ISSN 1548-0992.

JAIN, R.; PAUL, S. Network virtualization and software defined networking for cloud computing: a survey. *IEEE Communications Magazine*, v. 51, n. 11, p. 24–31, November 2013. ISSN 0163-6804.

KHAN; SAHRISH; WAHID; ABDUL; TANVIR; SADAF. Comparative study of routing strategies in software defined networking. In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2016. (SAC '16), p. 696–702. ISBN 978-1-4503-3739-7. Disponível em: <http://doi.acm.org/10.1145/2851613.2851853>.

KIM, H.; FEAMSTER, G. I. o. T. N. Improving network management with software defined networking. In: *IEEE Communications Magazine*. [s.n.], 2013. Disponível em: <http://gtnoise.net/papers/2013/kim-ieee2013.pdf>.

KLJAIC, Z.; SKORPUT, P.; AMIN, N. The challenge of cellular cooperative its services based on 5g communications technology. In: *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2016 - Proceedings*. [S.l.: s.n.], 2016. p. 1587–594.

KREUTZ, D.; RAMOS, F. M. V.; VERíSSIMO, P. E.; ROTHENBERG, C. E.; AZODOLMOLKY, S.; UHLIG, S. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, v. 103, n. 1, p. 14–76, Jan 2015. ISSN 0018-9219.

KREUTZ, D.; RAMOS, F. M. V.; VERíSSIMO, P. E.; ROTHENBERG, C. E.; AZODOLMOLKY, S.; UHLIG, S. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, v. 103, n. 1, p. 14–76, Jan 2015. ISSN 0018-9219.

LEHRIG, S.; EIKERLING, H.; BECKER, S. Scalability, elasticity, and efficiency in cloud computing: A systematic literature review of definitions and metrics. In: *2015 11th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA)*. [S.l.: s.n.], 2015. p. 83–92.

LI, J.; YOO, J. H.; HONG, J. W. K. Cpman: Adaptive control plane management for software-defined networks. In: *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. [S.l.: s.n.], 2015. p. 121–127.

LIANG, W.; LI, Z.; ZHANG, H.; WANG, S.; BIE, R. Vehicular ad hoc networks: Architectures, research issues, methodologies, challenges, and trends. *International Journal of Distributed Sensor Networks*, 2015. Disponível em: <http://journals.sagepub.com/doi/full/10.1155/2015/745303>.

MENDONCA; MARC; OBRACZKA; KATIA; TURLETTI; THIERRY. The case for software-defined networking in heterogeneous networked environments. In: *Proceedings of the 2012 ACM Conference on CoNEXT Student Workshop*. New York, NY, USA: ACM, 2012. (CoNEXT Student '12), p. 59–60. ISBN 978-1-4503-1779-5. Disponível em: <http://doi.acm.org/10.1145/2413247.2413283>.

MUNOZ, R.; VILALTA, R.; CASELLAS, R.; MARTINEZ, R.; SZYRKOWIEC, T.; AUTENRIETH, A.; LOPEZ, V.; LOPEZ, D. Integrated sdn/nfv management and orchestration architecture for dynamic deployment of virtual sdn control instances for virtual tenant networks [invited]. *IEEE/OSA Journal of Optical Communications and Networking*, v. 7, n. 11, p. B62–B70, November 2015. ISSN 1943-0620.

MUñOZ, R.; VILALTA, R.; CASELLAS, R.; MARTíNEZ, R.; SZYRKOWIEC, T.; AUTENRIETH, A.; LóPEZ, V.; LóPEZ, D. Sdn/nfv orchestration for dynamic deployment of virtual sdn controllers as vnf for multi-tenant optical networks. In: *2015 Optical Fiber Communications Conference and Exhibition (OFC)*. [S.l.: s.n.], 2015. p. 1–3.

NAOHISA, M.; KENKI, T.; SHO, H.; HIROKI, A. A. A. Iot network implemented with nfv. Jul 2016. Disponível em: <http://www.nec.com/en/global/techrep/journal/g15/n03/pdf/150308.pdf>.

NIKOS, B.; KUIPERS; A., F. SDN and Virtualization Solutions for the Internet of Things: A Survey. *IEEE Access*, v. 4, p. 5591–5606, 2016. ISSN 2169-3536. Disponível em: <http://ieeexplore.ieee.org/document/7563828/>.

OLARIU, S.; WEIGLE, M. C. *Vehicular networks: from theory to practice*. [S.l.]: Crc Press, 2009.

ONF, O. N. F. *OpenFlow Switch Specification*. 2013.

ONFTR-526. *SApplying SDN Architecture to 5G Slicing*. 2016. Disponível em: <https://www.opennetworking.org/wp-content/uploads/2014/10/Applying_SDN_Architecture_to_5G_Slicing_TR-526.pdf>.

OPENMANO. *OpenMANO Project*. 2014. Disponível em: <https://github.com/NFVLABS/OPENMANO>.

OPENSTACK. *Documentação Oficial Openstack*. 2018.

OPNFV. *OPNFV Project*. 2014.

OSRG. *Ryu SDN Framework*. 2017.

REZENDE, C.; MAMMERI, A.; BOUKERCHE, A.; LOUREIRO, A. A. F. A receiver-based video dissemination solution for vehicular networks with content transmissions decoupled from relay node selection. *Ad Hoc Netw.*, Elsevier Science Publishers B. V., v. 17, p. 1–17, jun. 2014. ISSN 1570-8705. Disponível em: <http://dx.doi.org/10.1016/j.adhoc.2013.12.011>.

RIZZO, G.; PALATTELLA, M. R.; BRAUN, T.; ENGEL, T. Content and context aware strategies for qos support in vanets. In: *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*. [S.l.: s.n.], 2016. p. 717–723. ISSN 1550-445X.

SAHASRABUDHE, S. S.; SONAWANI, S. S. Comparing openstack and vmware. In: *2014 International Conference on Advances in Electronics Computers and Communications*. [S.l.: s.n.], 2014. p. 1–4.

SALAHUDDIN, M. A.; AL-FUQAHA, A.; GUIZANI, M.; CHERKAOUI, S. Rsu cloud and its resource management in support of enhanced vehicular applications. In: *2014 IEEE Globecom Workshops (GC Wkshps)*. [S.l.: s.n.], 2014. p. 127–132. ISSN 2166-0077.

SPAHO, E.; BAROLLI, L.; MINO, G.; XHAFA, F.; KOLICI, V. Vanet simulators: A survey on mobility and routing protocols. In: *2011 International Conference on Broadband and Wireless Computing, Communication and Applications*. [S.l.: s.n.], 2011. p. 1–10.

SUDHEERA, K. L. K.; MA, M.; ALI, G. G. M. N.; CHONG, P. H. J. Delay efficient software defined networking based architecture for vehicular networks. In: *2016 IEEE International Conference on Communication Systems (ICCS)*. [S.l.: s.n.], 2016. p. 1–6.

TACKER. *Tacker Documentationt*. 2016.

TRUONG, N. B.; LEE, G. M.; GHAMRI-DOUDANE, Y. Software defined networking-based vehicular adhoc network with fog computing. In: *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. [S.l.: s.n.], 2015. p. 1202–1207. ISSN 1573-0077.

YAN, Q.; YU, F. R.; GONG, Q.; LI, J. Software-defined networking (SDN) and distributed denial of service (DDOS) attacks in cloud computing environments: A survey, some research issues, and challenges. *IEEE Communications Surveys and Tutorials*, v. 18, n. 1, p. 602–622, 2016. ISSN 1553877X. Disponível em: <http://ieeexplore.ieee.org/document/7289347/>.