UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
CURSO DE BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO

YTALLO GUSTAVO BRITO PESSOA

Avaliação de Desempenho de Redes Veiculares Ad Hoc(VANETs) definidas por Software

Recife
2018

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
CURSO DE BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO

YTALLO GUSTAVO BRITO PESSOA

Avaliação de Desempenho de Redes Veiculares Ad Hoc(VANETs) definidas por Software

Monografia apresentada ao Centro de Informática (CIN) da Universidade Federal de Pernambuco (UFPE), como requisito parcial para conclusão do Curso de Engenharia da Computação, orientada pelo professor Kelvin Lopes Dias.

RECIFE
2018

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
CURSO DE BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO

YTALLO GUSTAVO BRITO PESSOA

Avaliação de Desempenho de Redes Veiculares Ad Hoc(VANETs) definidas por Software

Monografia submetida ao corpo docente da Universidade Federal de Pernambuco, defendida e aprovada em 11 de Dezembro de 2018.

Banca Examinadora:

Orientador

Kelvin Lopes Dias
Doutor

Examinador

Jose Augusto Suruagy Monteiro
Doutor

Dedico este trabalho ao meu Deus que me fortaleceu durante esta jornada da Graduação. Também, aos meus pais, Manoel Paulino e Christiane Ledo, fundamentais em me incentivar a lutar pela vida.

AGRADECIMENTO

Ao Prof. Dr. Kelvin Lopes Dias pelo acompanhamento, direcionamento e paciência em me orientar de forma bastante solícita.

Ao doutorando Rhodney Arthur pelas dicas de pesquisa, tecnológicas e de implementação.

A todos os colegas de curso.

"Assim corro também eu, não sem meta; assim luto, não como desferindo golpes no ar."

(1 CORÍNTIOS 9:26)

RESUMO

As redes ad hoc veiculares (VANETs) vêm crescendo bastante no âmbito da pesquisa e no meio de produção. Há inúmeras aplicações suportadas por estas redes como serviços de segurança rodoviária, eficiência de tráfego e serviços de infoentretenimento. Porém, há inúmeros desafios no gerenciamento de recursos nestes tipos de redes, pois há uma alta dinâmica na mudança de topologia devido ao movimento constante dos carros. O paradigma de programação de redes definidas por software (SDN) é uma solução bastante atrativa para o gerenciamento das comunicações deste contexto. Nesta monografia avaliamos o desempenho de uma VANET baseada em SDN (Redes definidas por software), com suporte à mobilidade no controlador. Será utilizado, como principal ferramenta, o Mininet-Wifi, um emulador de redes sem fio definidas por software.

Palavras-Chave: VANETS, redes, SDN, Mininet-Wifi

ABSTRACT

Vehicular ad hoc networks (VANETs) have been growing significantly in research and production. There are numerous applications supported by these networks such as road safety services, traffic efficiency and infotainment services. However, there are innumerable challenges in resource management in these types of networks, as there is a high dynamics in the topology change due to the constant movement of the cars. The software programming paradigm (SDN) is a very attractive solution for the management of communications in this context. In this monograph we evaluated the performance of a VANET based on SDN (Software Defined Networks), with support for mobility in the controller. The main tool will be Mininet-Wifi, a wireless emulator defined by software.

Keywords: VANETS, networks, SDN, Mininet-Wifi

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

VANET - VEICULAR AD HOC NETWORKS

SDN - SOFTWARE DEFINED NETWORKING

XML - EXTENSIBLE MARKUP LANGUAGE

OSI – OPEN SYSTEM INTERCONNECTION

RSU – ROADSIDE UNIT

SUMO - SIMULATION OF URBAN MOBILITY.

IEEE - INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

Redes Veiculares Ad-Hoc (VANETs) vem recebendo um forte interesse por parte de pesquisadores e pela indústria automobilística. Existem diversas demandas, no que diz respeito a redes de automóveis extra veiculares. Segue algumas delas: carros autônomos, segurança nas estradas, melhor eficiência de tráfego e o *Intelligent Transport Services* (ITS). (KU *et al.*, 2014). Com isso, exigiu-se um avanço significativo no processo de comunicação de rede. Mas como ter uma rede que suporte as demandas das VANETs?

As redes de comunicações em geral, não só de computadores, estão passando por um processo de transição. As aplicações estão evoluindo rapidamente. Isso está exigindo um esquema mais rápido, sofisticado e “smart” (inteligente), no que diz respeito à comunicação entre hosts (dispositivos de rede). A rede vigente é quase que “*plug and play*”, onde os dispositivos de rede já vêm pré-programada, de acordo com as especificações técnicas genéricas e proprietárias. Analisando o modelo OSI de referência (Figura 1 abaixo), percebemos que pouco se avançou nas camadas 3,2 e 1. Uma rede baseada apenas em endereçamento IP pré-configurado, protocolos de roteamento e encaminhamento de camada 2 e 3 não suportam mais a complexidade das aplicações que estão surgindo atualmente.

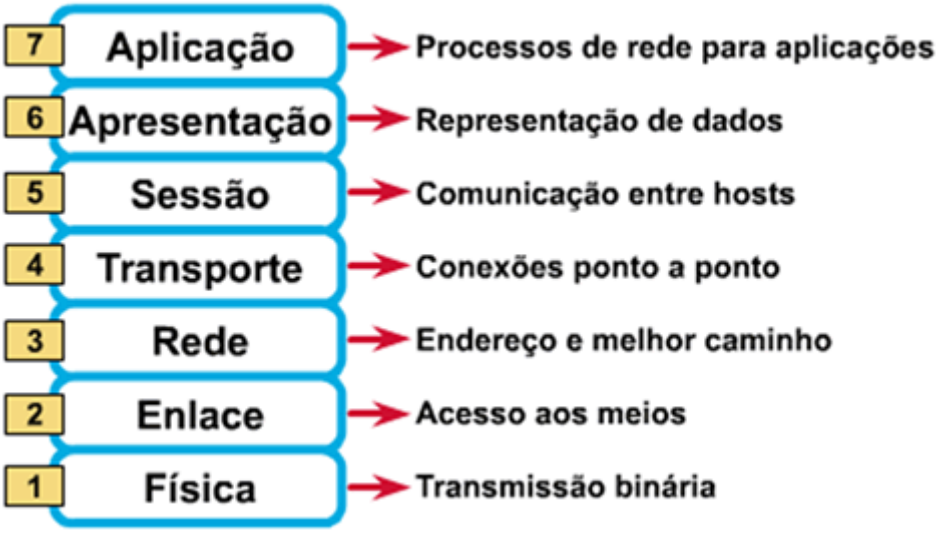

Figura 1 - Modelo OSI de Referência.
Fonte: Wikipedia

O controle e o encaminhamento de dados estão dentro do mesmo dispositivo destas camadas: roteadores e switches.

É aí que entra SDN (Software Defined Networking) ou redes definidas por software, visando desacoplar o Plano de Dados e o Plano de Controle, dando mais inteligência e flexibilidade às redes.

Podemos ver uma boa explicação de SDN em (TREVIZAN DE OLIVEIRA; BATISTA GABRIEL; BORGES MARGI, 2015, p. 01) que diz que "[...] Foi concebido com o objetivo de reduzir a complexidade de configuração e gerenciamento de redes, permitindo pesquisa e inovação em redes de produção.[...]".

Mas o que dá base para o funcionamento do SDN é o protocolo Openflow. Cada dispositivo da rede tem que ter suporte ao protocolo para poder ser inserido na hierarquia organizacional do SDN. O OpenFlow é de padrão aberto. Ele também cria uma camada de abstração para gerenciamento. Temos uma definição bem sucinta e objetiva em relação ao objetivo do protocolo OpenFlow em (CAMPOS; MARTINS, 2017):

> [...] Com o OpenFlow, os elementos de encaminhamento oferecem uma interface de programação simples que lhes permite estender o acesso e

> controle das tabelas de encaminhamento/roteamento utilizadas pelo hardware para determinar o próximo passo de cada pacote recebido.

O sucesso do uso do OpenFlow, é, também, por ele operar em diversos dispositivos, independentemente do fabricante. Se o dispositivo oferecer suporte a este protocolo, podemos simplesmente instalá-lo para operação na rede. Ele dita as regras de encaminhamento com base num microcontrolador central. Basicamente, o *OpenFlow* é um protocolo aberto de comunicação entre o controller e os equipamentos de redes que dão suporte ao mesmo. O protocolo vai atribuir inteligência a rede. Os equipamentos são previamente programados, onde atribui-se toda a lógica de encaminhamento para o destino.

1.2 Motivação

Os veículos estão cada vez mais equipados com sistemas embarcados, sensores, unidades de processamento e interfaces de comunicação sem fio. Assim, um leque de opções se abriu, no que diz respeito à aplicações de segurança, conforto e colaboração pública, ao mesmo tempo que os veículos se deslocam pela estrada.

As VANETs já são um fato e requerem muitos novos serviços e complexidade de rede. Muitas demandas são exigidas que vão desde entretenimento, novos serviços de rede; são utilizadas em carros autônomos, melhor controle de mobilidade urbana etc.

No entanto existem muitos desafios numa implementação desta rede. Podemos citar o tráfego de fluxo desbalanceado entre topologia de múltiplos caminhos e utilização de rede ineficiente. Com isso, requer-se uma arquitetura de rede veicular aberta e flexível como pré-requisitos importantes para dar liberdade para que os pesquisadores façam seus testes em um ambiente de produção, bem como para melhorar o gerenciamento de recursos de rede, aplicativos e usuários. (KU *et al.*, 2014)

Para resolver esta questão, utilizaremos os elementos e conceitos das Redes SDN que oferecem a flexibilidade e o gerenciamento requerido.

1.3 Problema

Tudo começou na rede tradicional (padrão IEEE 802.3) ou cabeada onde há um forte atrelamento entre hardware e software com todos os recursos pré-definidos. Depois veio a rede sem fio (padrão IEEE 802.11) para dar mais mobilidade aos nós sem fio. Baseado nessa ideia, chegamos às redes VANETs, onde os nós são carros e os Access Point são RSUs. Vejamos a Figura 2 para melhor entendimento e contextualização do problema a ser abordado:

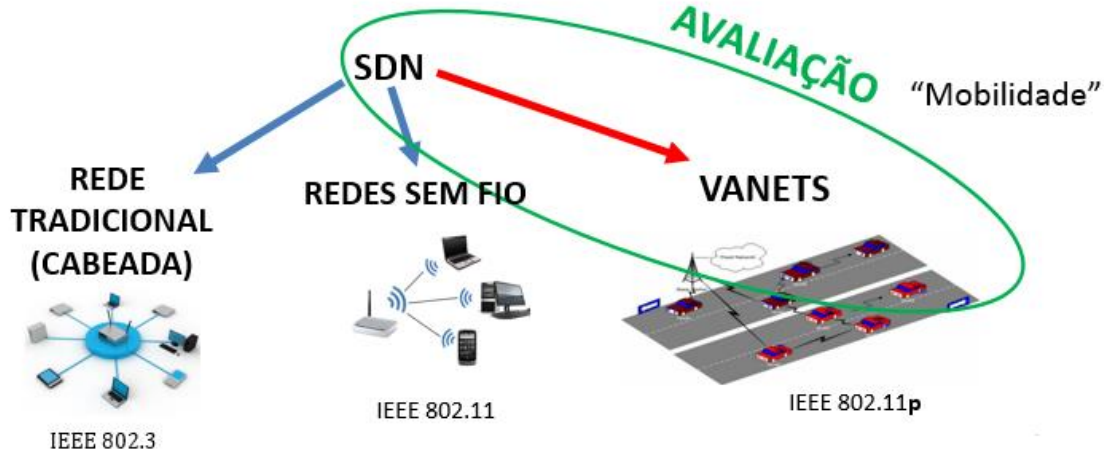

Figura 2 - Ilustração do problema a ser abordado

Conforme mostrado na figura, a SDN foi criada, inicialmente, para as redes tradicionais, mas foram implementadas soluções para WIFI. Posteriormente, ainda em estudos iniciais, SDN vem sendo aplicada às VANETs. Neste contexto de integração de VANETs e SDN é que se concentra nosso trabalho, avaliando o desempenho. Esta avaliação terá como principal diferencial o aspecto da mobilidade dos nós(veículos).

O artigo mais recente que usa as tecnologias abordadas neste trabalho, o (DOS REIS FONTES *et al.*, 2017) não trata da mobilidade dos nós, porque os autores quiseram manter o controlador Padrão do mininet. O controlador padrão não suporta handoffs(troca de RSUs). Mais pra frente iremos mostrar que utilizamos um controlador externo que dá suporte à mobilidade.

1.4 Objetivos

Daqui em diante o restante desta monografia tratará dos seguintes objetivos gerais:

a) Implementar SDN em VANETs
b) Avaliar o desempenho das redes VANETs no cenário proposto.

Este trabalho está organizado da seguinte maneira. A seção 2 apresenta mais os conceitos básicos. A seção 3 apresenta as características e arquitetura das redes VANETs. A seção 4 discute as ferramentas utilizadas e integração das mesmas. A Seção 5 debate alguns alguns trabalhos relacionados a fim de termos um panorama do estado da arte. A Seção 6 mostra a metodologia dos experimentos analisando-os. Finalmente, a Seção 7 conclui este trabalho e apresenta algumas orientações futuras.

2 CONCEITOS BÁSICOS

Nesta seção, iremos introduzir alguns termos e conceitos utilizados ao longo deste trabalho.

2.1 Redes sem fio (padrão IEEE 802.11)

O padrão IEEE 802.11 define a comunicação, codificação e transmissão dos dados em redes sem fio. Características do meio físico, frequência de rádio e infravermelho também são definidos neste padrão. O padrão original foi lançado em 1997. Posteriormente ocorreram inúmeras revisões e avanços anuais.
A área coberta pela rede sem fio é seccionada em células ou também chamadas de BSA(*Basic Service Area*). BSS(*Basic Service Set*) são estações que estão interagindo em uma BSA. Segue a Figura 3, exemplo de uma rede Ad-Hoc, para um melhor entendimento:

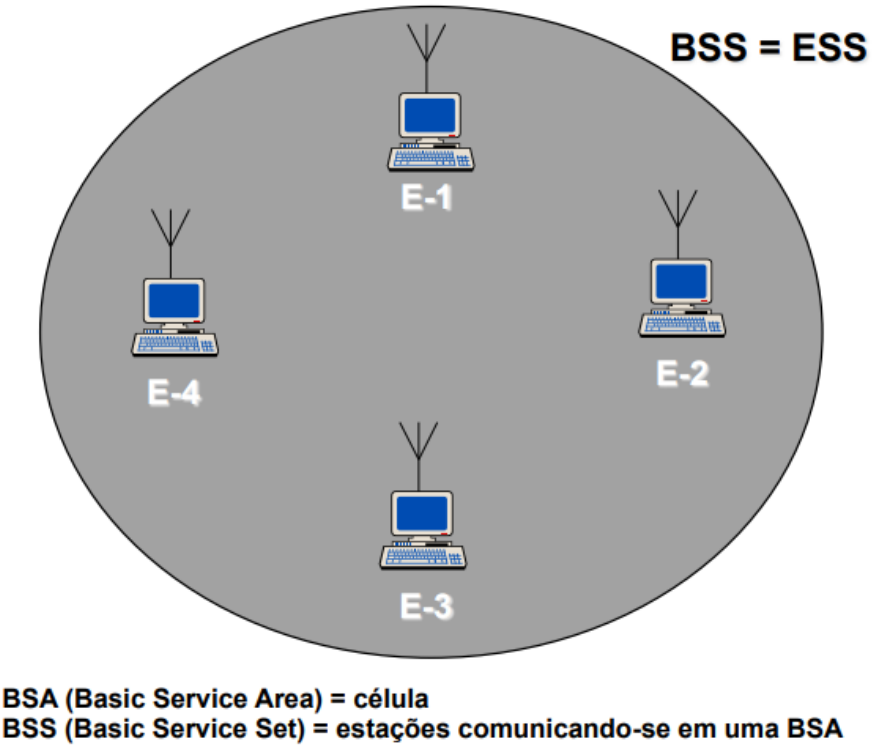

Figura 3 - rede Ad-Hoc

Fonte:http://www.midiacom.uff.br/debora/images/disciplinas/2017-2/redes/parte6a-2.pdf, pag. 7. Acessado em Novembro de 2018.

Um outro elemento muito importante numa rede sem fio é o AP(Access Point). Basicamente, este elemento faz o gerenciamento das comunicações entre origem e destino. Veremos a Figura 4 para entendermos deste dispositivo:

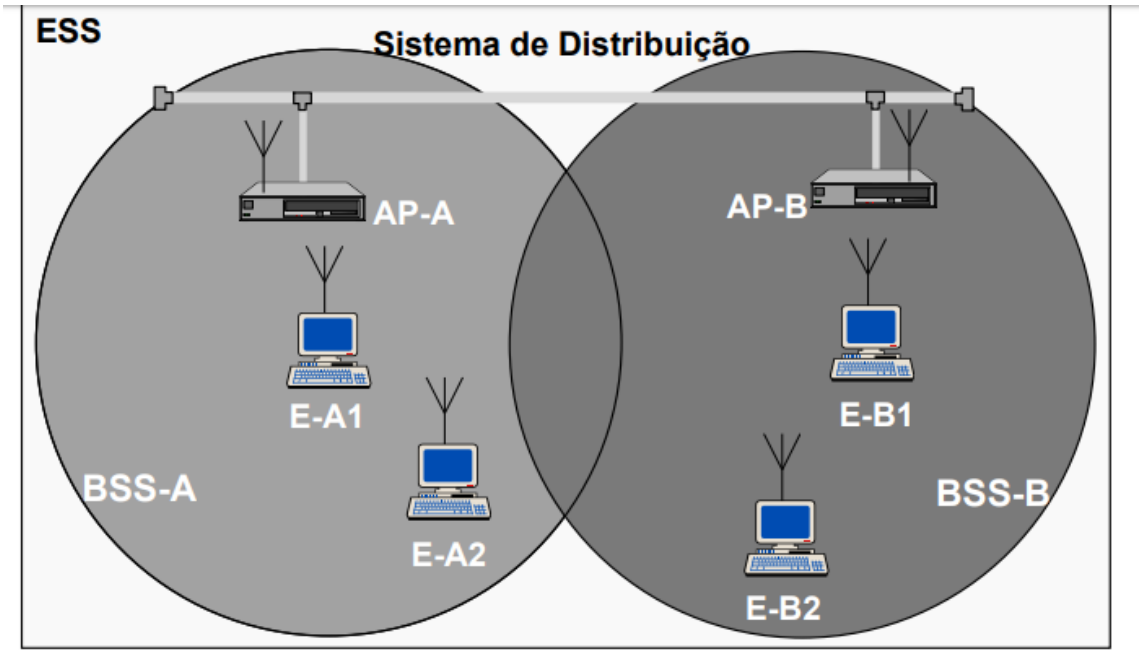

Figura 4 - Sistema de distribuição Wifi

Fonte:http://www.midiacom.uff.br/debora/images/disciplinas/2017-2/redes/parte6a-2.pdf, pag. 8. Acessado em Novembro de 2018.

As redes sem fio têm algumas limitações, pois é complicado detectar outro sinal além da estação de origem ou destino, no que se diz respeito às antenas. Outro problema é que algumas estações não permanecem ao alcance das outras.

Uma técnica de controle de colisão dos quadros em meio sem fio é o protocolo CSMA/CA.

Outro ponto importante das redes sem fio é conhecermos a estrutura e os campos do quadro MAC 802.11

A seguir segue a explicação de cada campo:

- Frame control: Versão do protocolo (0)
- Tipo do quadro (gerência, controle, dados) Se quadro foi fragmentado
- DS bits (Distribution System):
- Significado dos 4 endereços MAC

- Duration/ID – período de tempo em que o meio ficará ocupado ou Association ID da estação (PS poll)
- Sequence control – número de fragmento/sequência para reconhecer quadros duplicados (na falta de ACK)
- CRC – detecção de erro CRC-32

2.2 Redes SDN (*Software-Defined Network*)

SDN(Software Defined Networking) ou redes definidas por software, visa desacoplar o Plano de Dados e o Plano de Controle, dando mais inteligência e flexibilidade às redes.
Podemos ver uma boa explicação da concessão de SDN em (TREVIZAN DE OLIVEIRA; BATISTA GABRIEL; BORGES MARGI, 2015, p. 01) que diz que "[...] foi concebido com o objetivo de reduzir a complexidade de configuração e gerenciamento de redes, permitindo pesquisa e inovação em redes de produção.[...]".

Com essa nova abordagem de redes, podemos isolar o processo de decisão de encaminhamento em outro ponto da rede chamado de Controlador SDN. É nele onde fica toda a inteligência e concentração de filtros de encaminhamento, como na Figura 5 abaixo:

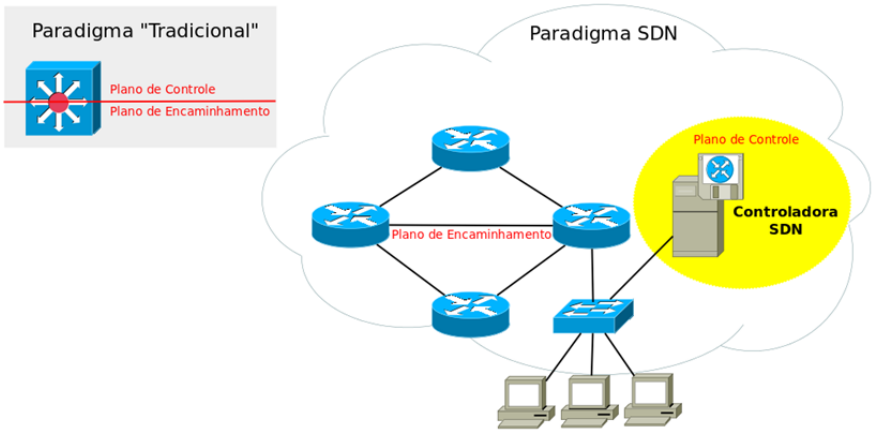

Figura 5 - Arquitetura de redes definidas por software

Fonte: http://labcisco.blogspot.com.br/search?q=SDN/s1600/SDN.png. Acessado em Novembro de 2017.

Mas o que dá a base para o funcionamento do SDN é o protocolo Openflow. Cada dispositivo da rede tem que ter suporte ao protocolo para poder ser inserido na hierarquia organizacional do SDN. O OpenFlow é de padrão aberto. Ele também cria uma camada de abstração para gerenciamento. Temos uma definição bem sucinta e objetiva em relação ao objetivo do protocolo OpenFlow em (CAMPOS; MARTINS, 2017):

> [...] Com o OpenFlow, os elementos de encaminhamento oferecem uma interface de programação simples que lhes permite estender o acesso e controle das tabelas de encaminhamento/roteamento utilizadas pelo hardware para determinar o próximo passo de cada pacote recebido.

O sucesso dos inúmeros usos do OpenFlow, é, também, por ele operar em diversos dispositivos, independentemente do fabricante. Se o dispositivo oferecer suporte a este protocolo, podemos simplesmente instalá-lo para operação na rede. Ele dita as regras de encaminhamento com base num microcontrolador central. Basicamente, o OpenFlow é um protocolo aberto de comunicação entre o *controller* e os equipamentos de redes que dão suporte ao mesmo. O protocolo vai atribuir inteligência a rede. Vejamos mais detalhes na Figura 6:

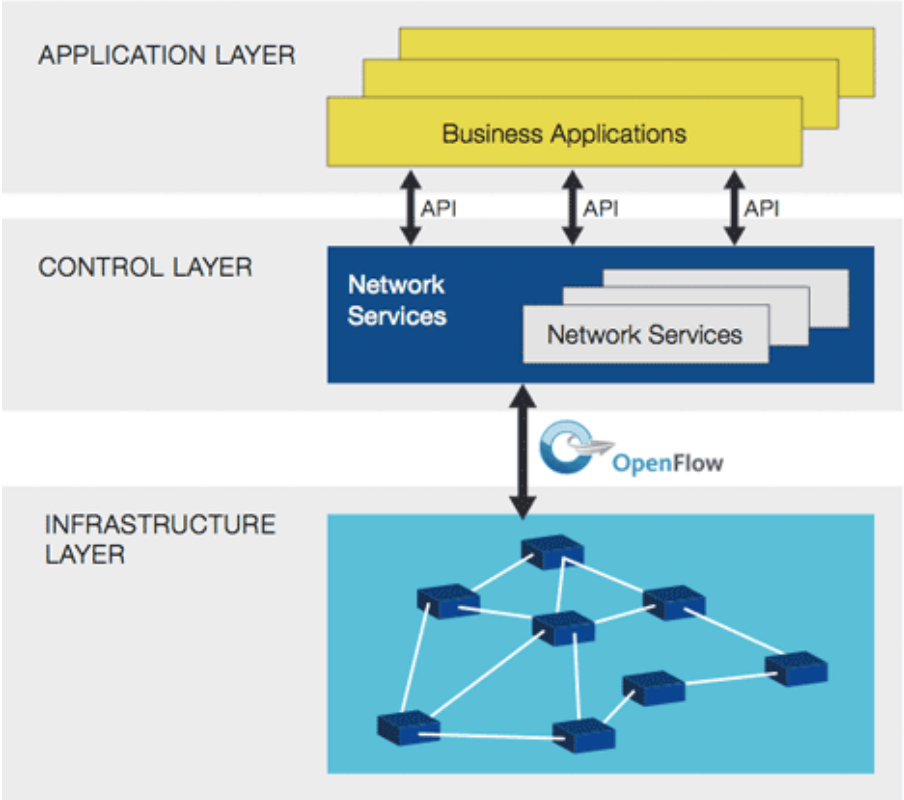

Figura 6 - Arquitetura SDN em camadas

Fonte: www.static.computerworld, 2018. Acessado em Novembro de 2018.

Os equipamentos são previamente programados, onde atribui-se toda a lógica de encaminhamento para o destino. Já o *FlowVisor* é um modo de virtualizar arquitetura de redes OpenFlow. Virtualização de Redes é um importante conceito estudado atualmente. Com a virtualização, abre-se a possibilidade de criar várias máquinas virtuais num mesmo hardware para diferentes fins. Os recursos de hardware precisam ser divididos. O OpenFlow é um exemplo de virtualização, pois permite, por exemplo, que diferentes tipos de fluxos utilizem, concomitantemente, na mesma rede, os recursos da infraestrutura divididos pelo FlowVisor, como segue um exemplo para melhor entendimento na Figura 4 abaixo:

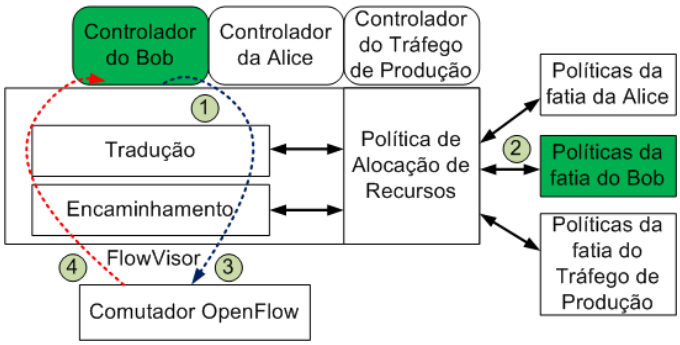

Figura 7 - Exemplo de uso do FlowVisor

Fonte: Disponível em https://www.gta.ufrj.br//flowvisor.html. Acessado em Novembro de 2018.

2.2.1 Controlador SDN

O controlador SDN é basicamente o “sistema operacional” da rede. Hoje em dia o controlador utiliza o OpenFlow pra implementar as regras de escalonamento, barreira e outros parâmetros referentes às regras e questão. Então o controlador cria uma camada de abstração para as aplicações que irão rodar em cima da rede. Em baixo da abstração ficam os equipamentos ou dispositivos de rede. SDN, através de seus controladores, trouxe de volta o comando da rede para o administrador de rede. Antes as regras de rede ficaram engessadas nos dispositivos proprietários que vinham com políticas pré-instaladas. Por isso SDN ganhou muita visibilidade e é tão comentada e discutida no âmbito acadêmico e corporativo. Agora o operador de rede não fica mais limitado ao que o hardware de rede faz. Se o hardware de rede não faz o que queremos, podemos implementar isso, a especificidade requerida, dentro do controlador. Os donos das redes deixam de ser as grandes empresas dos dispositivos. Nos controladores também têm a possiblidade de escrevermos regras de firewall sem ser no link comum de instalação física de um firewall de rede. Estas novas regras podem ser aplicadas na rede inteira de uma só vez, e não apenas concentrar num ponto da rede, como é o comum nas redes tradicionais. O link estático da rede, deixa de ser um lugar estratégico para diversas questões de segurança. Vejamos a Figura 8 ilustrativa que mostra um controlador SDN situado na rede

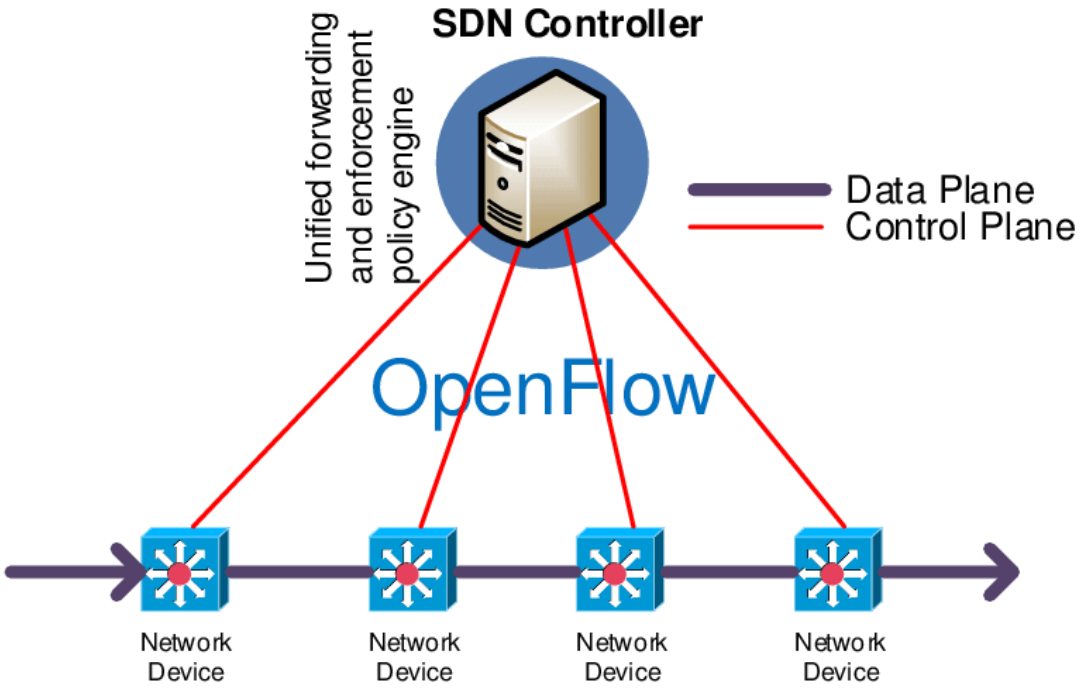

Figura 8 - Controlador SDN situado na rede

Fonte: Disponível em https://www.researchgate.net/implemented-with-OpenFlow.png . Acessado em Novembro de 2018.

É no controlador que também tratamos os pacotes em no que diz respeito ao encaminhamento dos mesmos. Se o pacote bate com uma entrada da tabela de encaminhamento dos dispositivos, o mesmo entra numa lógica algorítmica de encaminhamento, fazendo o que o programador da rede descreveu que pode ser reescrita de cabeçalhos, reencaminhamento, bloqueio, descarte de pacote, inspeção, cópia, recuperação ou correção de erros de transmissão.

3 REDES VANETS

Nesta seção veremos os aspectos principais das redes VANETs, que são a base para este trabalho.

3.1 Visão Geral

Redes Veiculares Ad-Hoc (VANETs) oferecem uma gama de serviços de rede nas estradas, que vão desde segurança, apoio aos condutores de veículos e entretenimento. Estas redes têm veículos munidos de sensores integrados, processadores, dispositivos de armazenamento, além de placas de comunicação sem fio, onde os veículos podem se comunicar entre si (Ad-Hoc) ou estabelecer conexão com estações base que são distribuídas pela rede. (COSTA *et al.*, 2016).

Distribuir conteúdo(dados) eficientemente com o mínimo de atraso e overhead, e, também, uma boa área de cobertura, não é simples, pois, VANETs tem um alto grau de dinamicidade e falhas de comunicação V2V devido a alta mobilidade dos veículos. (SILVA *et al.*, 2016)

Numa VANET, existem, basicamente, dois tipos de comunicação. São elas:

1. V2V(vehicle-to-vehicle: os carros se comunicam entre si sem precisar de um suporte de infraestrutura estático.
2. V2I(vehicle-to-infrastructure): os carros se comunicam com RSUs(road-side-units), tipo de AP que são colocados ao o longo da estrada pra gerenciar a comunicação na rede.
3. Arquitetura híbrida: comunicação entre todos os dispositivos que não são carros(estações móveis).

Segue a Figura 9 que ilustra bem essas comunicações:

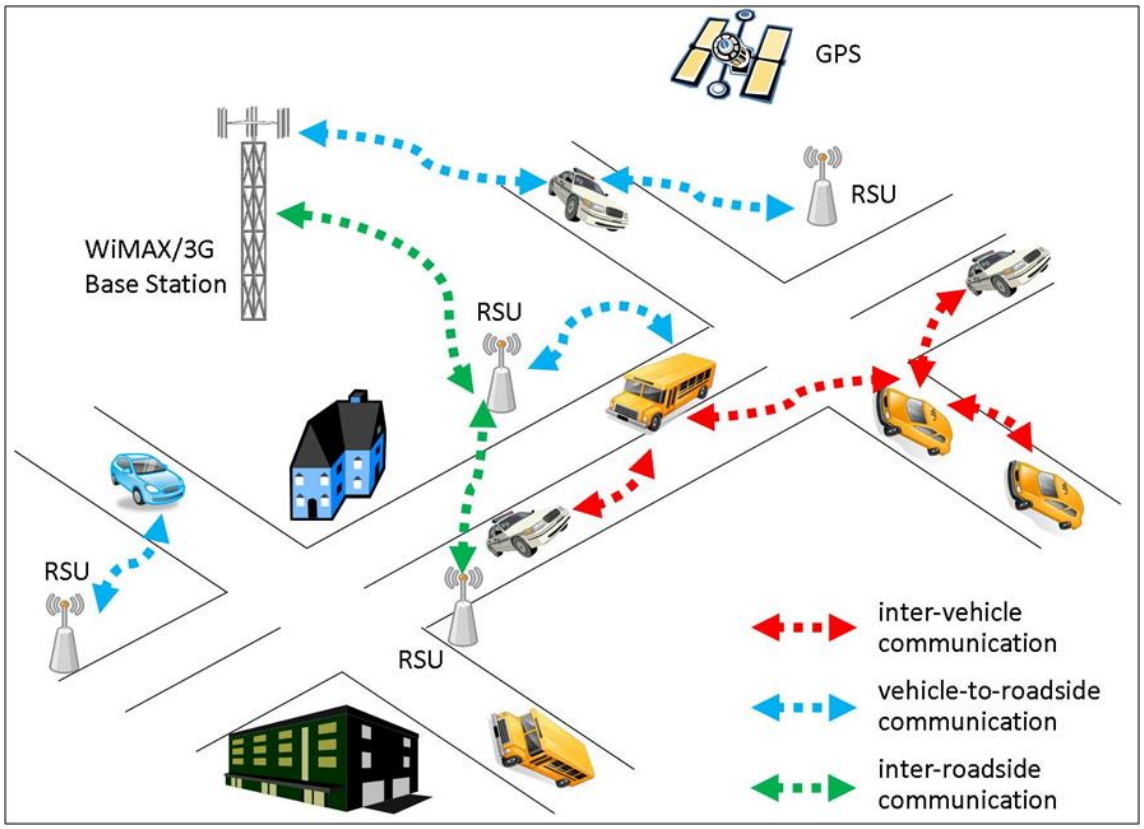

Figura 9 - Componentes e comunicações VANET

Fonte: https://ieeexplore.ieee.org/document/6849111. Acessado em Novembro de 2018.

3.2 Caraterísticas especiais

As VANETs têm aspectos peculiares, além da baixa largura de banda, curto alcance de transmissão e transmissão omnidirecional. Esses aspectos são formulados em (CUNHA *et al.*, 2016). Resumidamente, são eles:

- **Topologia dinâmica**: a velocidade é o principal motivo por esse dinamismo. Não podemos esquecer que a propagação de rádio também interfere nesta questão, pois os veículos se deslocam em diferentes direções.
- **Frequência de desconexão em comunicação V2V**: dois veículos podem rapidamente perder conexão enquanto se movimentam e transmitem informações.

- **Comunicação Geográfica**: em outras redes, os veículos(estações) são acessados por uma ID de área ou grupo. No caso das VANETs, regularmente dependem da sua localização geográfica.
- **Mobilidade e previsão limitadas**: apesar da topologia ser altamente dinâmica, as VANETs têm modelos de mobilidades bem definidos, pois seguem as restrições e elementos de trânsito. Dado isso, a posição futura do veículo é mais fácil de se prever.
- **Modelo de propagação**: interferência de reflexão e atenuação do sinal podem ser obstáculos nas redes VANETs. Isso se agrava no meio urbano, onde temos alta densidade de edificações.

3.3 Aplicações

As aplicações veiculares estão inseridas no que chamamos de Intelligent Transportation System (ITS), que tem a finalidade de melhorar a segurança e conscientização urbana por tecnologia.(CUNHA *et al.*, 2016). Porém, um pré-requisito importante pra que ITS funcione, a contento, são as VANETs.

VANETs oferecem serviços de segurança, que têm o objetivo de fornecer segurança dos veículos nas estradas, na tentativa de evitar acidentes. O gerenciamento de tráfego fornece informações dos locais atuais e configurações de dados para os mapas digitais. Já os serviços de entretenimento, têm o objetivo de transferência de dados multimídia. (KU *et al.*, 2014).

Outras aplicações mais específicas de ITS também podem ser citadas, tais como em (CUNHA *et al.*, 2016):

> [...] como a vigilância civil (fotos tiradas de cenas de violência em andamento enviadas a autoridades públicas por meio de infra-estrutura), controle de poluição, planejamento de estradas e tráfego e inúmeras outras aplicações com consciência urbana. Finalmente, meios mais agradáveis para fornecer acesso à Internet, informações turísticas / de publicidade, mídia social na estrada, orientação para as pessoas seguirem umas às outras na estrada, jogos, downloads de arquivos e aplicativos sociais (por exemplo, microblogs e chats)"

Em (CUNHA *et al.*, 2016) temos uma tabela que classifica as aplicações de VANET e suas características:

Tabela 1 - Classificação das aplicações VANET

Categorization of VANETs applications.

Application Class	Characteristic to consider	Architecture	Location awareness	Time awareness	Communication technology	Desirable properties of protocols	Challenges
Safety	Delay	V2V–V2I	Yes	Yes	DSRC/RFID/ Bluetooth/Wi-Fi	Reliability	Reduce the latency
Efficiency	Availability	V2V–V2I	Yes	Yes	DSRC	Real-time and reliability	Availability Of services
Comfort	Reliability	V2I	Yes	No	WiMAX/Wi-Fi/ 3G/4G/LTE	Real time	Support on-demand applications
Interactive Entertainment	Connectivity and availability	V2V–V2I	No	Yes	3G/LTE/Wi-Fi/ WiMAX	Unicast Communication	Keep synchronization
Urban Sensing	Mobility	V2V–V2I	Yes	Yes	DSRC/3G/LTE/ Wi-Fi/WiMAX	Data collection	Security in data communication

Fonte: retirada de (CUNHA *et al.*, 2016).

3.4 Redes VANETs definida por software

Redes VANETs definida por software(ou VANETs com SDN) é uma arquitetura que estende o conceito de SDN para uma rede ad hoc sem fio móvel entre veículos e infraestrutura correspondente.

Nesta seção descreveremos como podemos utilizar o conceito e técnicas de SDN para melhorar a flexibilidade e a utilização dos recursos das redes VANETs.

3.4.1 Componentes

(KU *et al.*, 2014) define os componentes da VANET em Controlador SDN, NÓ SEM FIO SDN e SDN RSU. O primeiro, Controlador SDN, é responsável por controlar todos os dispositivos com suporte à configuração do plano de controle, gerenciando todo o sistema. O segundo, NÓ SEM FIO SDN, são elementos que estão inseridos no plano de dados, recebendo mensagens de controle do CONTROLADOR SDN. Já o terceiro, SDN RSU, é um elemento estacionário, contido no plano de dados que podem ser controlados pelo controlador. Estes são distribuídos ao longo da estrada de rodagem.

Na figura 10 abaixo, exibe os componentes de um nó sem SDN. É bastante similar a um switch que tem suporte a openflow.

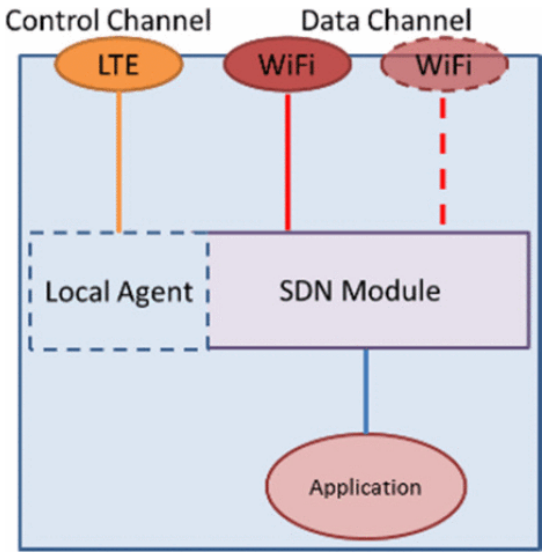

Figura 10 - Componentes de um nó sem SDN

Fonte: Adaptado de (KU *et al.*, 2014)

Um nó SDN possui um agente local que fará a o comunicação e sincronismo com o controlador SDN. Todos os recursos que estarão em funcionamento dependerão dos ativos no nó. Os nós podem atuar com hosts para encaminhamento e envio/recebimento de tráfego, característica de uma rede Ad hoc.

A diferença de uma SDN tradicional para uma SDN VANETs, é que esta última pode operar em 3 modos de controle: Central, Distribuído e Híbrido. No primeiro, o Central, o controlador governa todos os nós sem fio e as RSUs. O segundo, Distribuído, RSUs e nós subjacentes não operam durante a entrega dos pacotes de dados, como uma rede auto-organizada. O terceiro, Híbrido, o controlador tem controle total/nenhum, em alguma região de gerenciamento.

4 FERRAMENTAS E TECNOLOGIAS UTILIZADAS

Nosso objetivo é avaliar o desempenho das redes VANETs definidas por software. Porém, para termos resultados mais próximos dos reais, utilizaremos “emuladores” e não “simuladores”. As ferramentas apresentadas nesta seção, nos darão os subsídios necessários.

4.1 Controlador RYU

O RYU é um controlador SDN baseado em componentes de software com uma API bem documentada e desenvolvida. Isto pode torna mais produtivo a produção de softwares que utilizam esse controlador, o gerenciamento de novos aplicativos. Esta abordagem facilitou o crescimento da comunidade desenvolvedora. Ele também suporta vários protocolos para gerenciamento de redes SDN, tais como: NETCONF, OF-CONFIG, etc.

O controlador RYU ganhou uma grande visibilidade por ser totalmente escrito em PYTHON, e também por suportar as versões mais novas do OpenFlow 1.0, 1.2, 1.3 ,1.4 e 1.5. (FERNANDES; ROTHENBERG, 2014). Tem uma qualidade que já foi integrada com o OpenStack, no que diz respeito a virtualização de redes na nuvem. Com isso este controlador pode programar os switches pra criar VLANs de isolamento dentro da rede. O RYU tem um tempo de aprendizagem moderado. Utilizando o as versões mais novas do Linux, podemos instalá-lo utilizado a linha de comando abaixo no console do sistema operacional:

Linux $ pip install ryu

Segue a Figura 5 que mostra como o controlador se articula na rede SDN em produção:

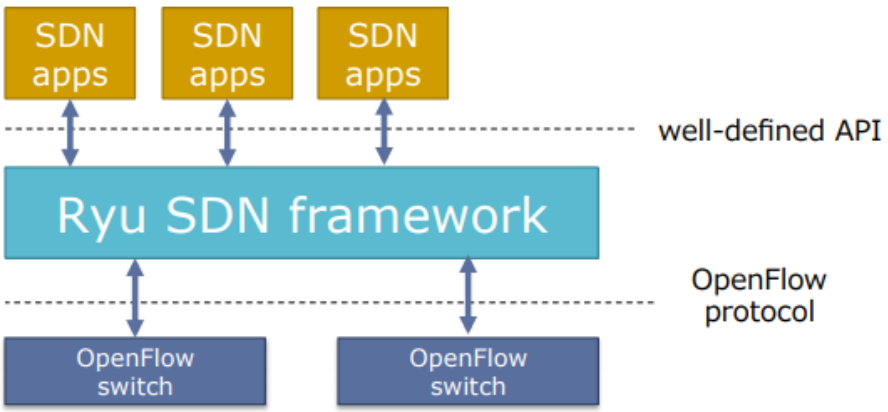

Figura 11 - Arquitetura Controlador RYU

Fonte:Disponível em <https://www.sdxcentral.com/wp-content/uploads/2014/09/ryu-controller-sdn-framework.jpg>. Acessado em Dezembro de 2017.

Atualmente o Ryu pode ser utilizado com python 2.7 e 3.4. Está sendo distribuído sob Apache License V2.0. Neste link < https://sourceforge.net/projects/ryu/> podemos visualizar as versões atualizações do projeto da comunidade de desenvolvimento.

4.2 OpenStreetMap(OMS)

É um projeto que foi criado em 2004 por um americano chamado Steve Coast. Esta aplicação tem por objetivo tornar livre os dados geoespaciais para a produção de mapas. O OpenStreetMap pode ser acessa do em www.openstreetmap.org.

Utilizamos esta tecnologia para poder gerar um mapa georreferenciado, onde posteriormente integraremos com o SUMO. A Figura 12 mostra como escolhemos a cidade de New York para fazermos os experimentos da seção 6.

Figura 12 - Região de Análise no OpenStreetMap

4.3 SUMO

SUMO é o acrônimo para Simulação de Mobilidade Urbana. É um software de código aberto que simula o tráfego rodoviário. A Figura 12 mostra a o ambiente GUI deste software:

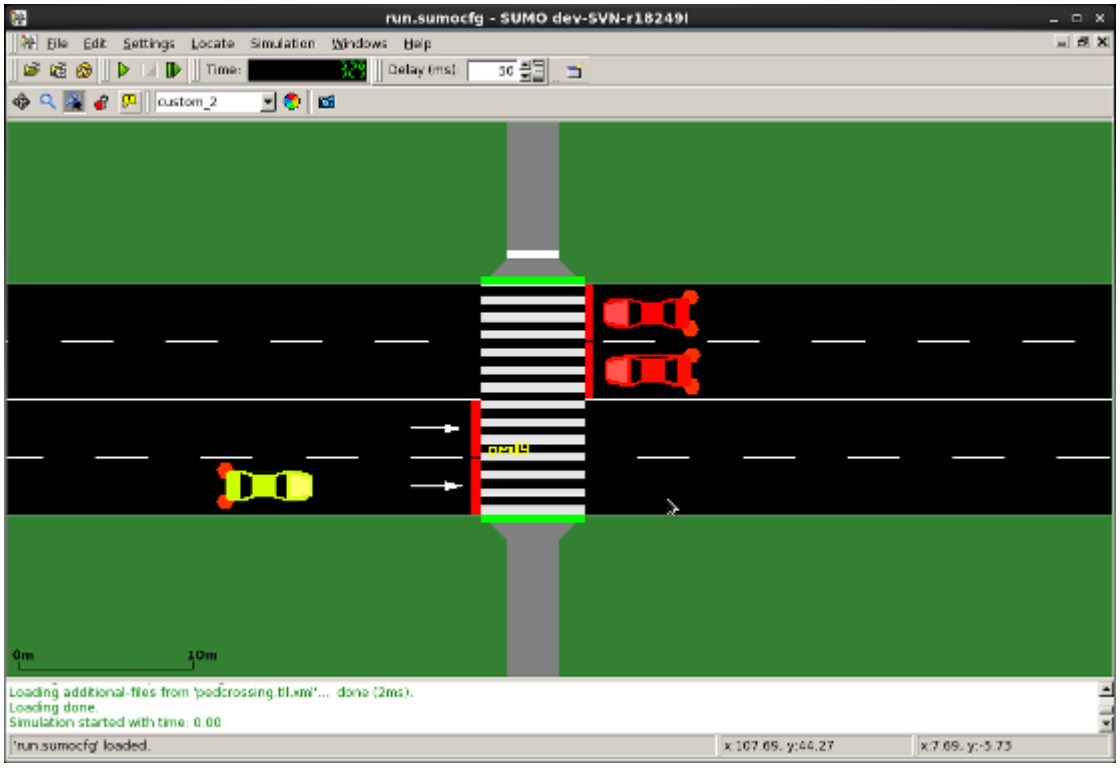

Figura 13 - Tela do SUMO

A sua documentação pode ser encontrada em http://sumo.dlr.de/wiki.
Para instalar o SUMO em ambiente Linux, basta fazer:

```
sudo add-apt-repository ppa:sumo/stable
sudo apt-get update
sudo apt-get install sumo sumo-tools sumo-doc
```

4.4 Python

Python é uma linguagem de programação de alto nível para propósitos gerais. Foi lançada por Guido VAN Rossum em 1991.
Nela, existem muitas bibliotecas relacionada a redes de comunicações. Esta linguagem foi utilizada para programar as classes que fizeram a modelagem da topologia, bem como os nós(estações e hosts) e elementos de redes.
Para instalar o python no Linux façamos:

```
$ sudo apt-get install python2.7
ou
$ sudo apt-get install python3.5
```

4.5 Open vSwitch (OVS)

Como utilizamos SDN, precisamos de um switch configurável que tivesse suporte ao *OpenFlow*. Para isto, utilizamos o OVS. OVS é um switch virtual, emulados sob plataforma Linux, multicamadas e de qualidade de produção.

A Figura 14 mostra as principais características deste switch virtual e configurável.

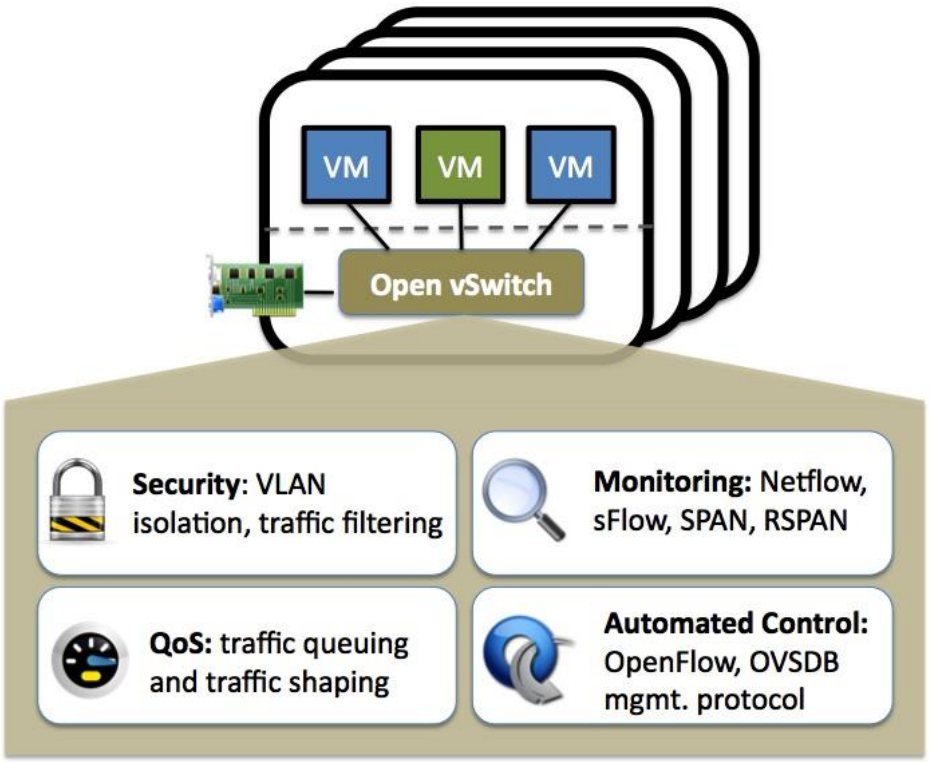

Figura 14 – Arquitetura do OpenVSwitch

A documentação pode ser acessada em http://www.openvswitch.org/download/.

Para baixar a última versão, durante a escrita desta monografia acesse:

```
http://openvswitch.org/releases/openvswitch-2.10.1.tar.gz
```

4.6 Mininet-Wifi

O Mininet-Wifi é uma extensão do famoso Mininet, bastante utilizado no meio acadêmico e de pesquisa. Ele é um emulador (imita o comportamento de um conjunto de hardware) de Redes Wifi Definidas por Software. A documentação e os códigos referentes à plataforma podem ser acessados em https://github.com/intrig-unicamp/mininet-wifi. Os criadores são Ramon dos Reis

Fontes e Christian Esteve Rothenberg da Universidade Estadual de Campinas(Unicamp).

O Mininet-Wifi tem funcionalidades novas tais como: estações wifi, pontos de acessos baseados no padrão Linux sem fio. Para instalar este emulador, faça:

```
sudo apt-get install git
git clone https://github.com/intrig-unicamp/mininet-wifi
cd mininet-wifi
sudo util/install.sh -Wlnfv
```

A Figura 14 mostra a arquitetura do Mininet-Wifi:

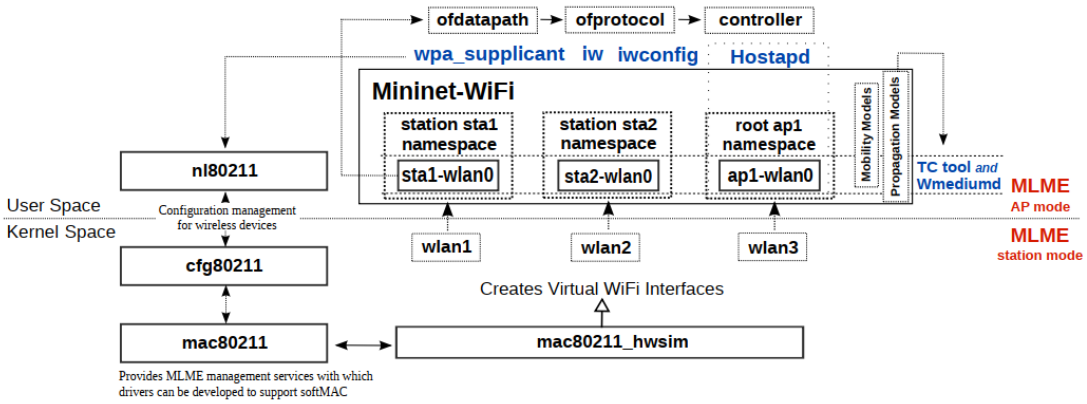

Figura 15 - Arquitetura do Mininet-Wifi

Basicamente, temos dois drivers Linux da arquitetura que são importantes para o nosso trabalho. Primeiro é driver linux ***mac80211_hwsim*** é responsável pela criação de interfaces Wi-Fi virtuais, importantes para estações e pontos de acesso. O segundo é o **Wmediumd** que é uma abordagem para emular a rede sem fio onde faz o controle, perda e transmissões de pacotes.

5 TRABALHOS RELACIONADOS (ESTADO DA ARTE)

Apresentaremos, nesta seção 5, os principais trabalhos relacionados ao nosso objeto de estudo: redes VANET.

5.1 Estudo dos desafios, protocolos e aplicativos.

Em (CUNHA *et al.*, 2016) são estudados as teóricos dos principais desafios e especificações da implementação de redes VANETs. Como é uma área de pesquisa extremamente nova, e, somado aos grandes avanços tecnológicos no setor automotivo, com sistemas embarcados, unidades de processamento mais avançadas e comunicação sem fio, diferentes protocolos de comunicação têm de ser desenvolvidos para suprir as especificidades destas novas redes. O trabalho foca na pesquisa de diferentes cenários, desafios, pilha de protocolos destes tipos de redes e uma análise comparativa entre os protocolos comuns. O problema deste *paper* é que não tem nenhum experimento real para sabermos de fato são os resultados de uma comunicação em uma VANET.

5.2 Sistemas de Gestão de Tráfego (TMS) em VANETs

Em (SOUSA; ARAÚJO; SAMPAIO, 2018) os autores propõem a chamada Redes Veiculares de Dados Nomeados (VNDN) para enfrentar os desafios das comunicações de rede VANET, como já foi falado na seção 2.3.2. Inundações de informações surgem em VANETs, pois há uma complexidade criada devido à mobilidade, dinamismo topológico e a velocidade dos veículos. Isto causa um certo grau de instabilidade pois não uma previsão satisfatória, tanto na posição do veículo como no tempo de conexão, ocasionando muitos handoffs. O artigo mostra uma estratégia de encaminhamento baseado em decisões de retransmissões. Porém, novamente, não há uma emulação de um conexão real para sabermos o que, de fato, acontece de retransmissão pra podermos comparar quantitativamente.

5.3 Protocolo de roteamento híbrido adaptativo para VANETs

Em (LIN *et al.*, 2017) novamente aborda os desafios da comunicação em uma VANET devido ao alto grau de mobilidade. Posteriormente, foca no design de

protocolos que atendam esta rede com características peculiares. Os autores mostram que como temos uma arquitetura híbrida, devemos ter protocolos híbridos. Para isto, eles projetam uma arquitetura baseada em “zona móvel”, que pretende disseminar a informação. As técnicas para modelar grandes bancos de dados de estação móveis para formular os protocolos híbridos. O problema deste paper é que não há informações de tráfegos reais de redes VANETs.

5.4 Gestão de Recursos em Redes Veiculares Definidas por Software

O artigo (DOS REIS FONTES *et al.*, 2017) será a nossa principal fonte pra elaboração deste trabalho. Ele começa tratando das dificuldades que é gerenciar recursos num ambiente tão dinâmico quanto é numa rede VANET. Depois, ele começa falando das aplicações e serviços fornecidos por estas redes. Daí, ele começa aplicando o conceito de SDN às VANETs como alternativa de programação aberta e proposta de melhora de desempenho. Porém, seu experimento se dá com o carro estático, não havendo portanto, políticas de mobilidade. Então, aí está o principal diferencial do nosso trabalho, avaliar o desempenho das VANETs baseadas em software com a mobilidade dos carros.

6 EXPERIMENTOS E ANÁLISE

6.1 Sobre a Simulação

O desempenho da rede VANET com SDN é avaliado usando o simulador SUMO integrado ao Mininet-Wifi, softwares utilizados nesta monografia já explicados na seção 3. Essas ferramentas foram utilizadas por serem de código aberto, pois são configuráveis, modificáveis e extensíveis. Também oferecem um gama de recursos.

A rede é configurada usando classes do Python, onde chamamos métodos de criação de nós, configuração de comunicações e etc. O código pode ser visualizado no Apêndice 1.

Utilizamos a ferramenta IPERF para estressar o link e fazer análise de desempenho. Este software faz a injeção de pacotes com vista a sobrecarregar o link e extrair estatísticas de desempenho.

A Figura 16 mostra o esquema que o IPERF trabalha:

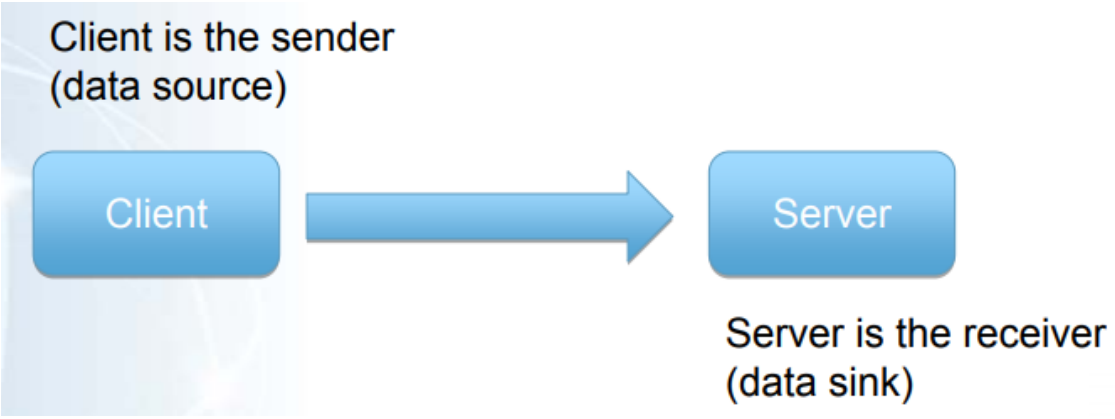

Figura 16 - Comunicação IPERF

Criamos um servidor de streaming que recebe/responde pacotes UDP vindo dos carros clientes. Optamos por usar o protocolo de transporte UDP, pois ele providencia uma maior transparência para sabermos o que tá acontecendo na rede, efetivamente. O TCP esconde informações de Perda, Jitter e a Entrega fora de ordem.

Dito isto, utilizando o IPERF, avaliamos as seguintes variáveis:

- **Vazão**: ou seja, o a velocidade efetiva do link, sobre uma Largura de Banda de 27Mbits/s, que é a capacidade especificada do protocolo IEEE802.11p para redes veiculares.
- **Jitter**: variação estatística do atraso.
- **Perda**: porcentagem dos pacotes perdidos em relação aos pacotes enviados.

Com relação ao mapa do SUMO, escolhemos a cidade de Nova Iorque, nos Estados Unidos. A razão da escolha, foi a boa base de trace automotivos, aumentando ainda mais o grau de realidade dos experimentos.

A Figura 17 mostra uma área de 20Km² que foi que representa o mapa escolhido:

Figura 17 - Imagem do Google Earth da Região de Tráfego dos Carros

Fonte: GOOGLE. Google Earth. Version 2018. ano. Nova York,EUA. Disponível em: <https://earth.app.goo.gl/KWB5aE>. Acesso em: 11 de Novembro de 2018.

6.2 Experimento 1: Análise do impacto da variação da velocidade

A seguir mostraremos os parâmetros fixos para simulação:

Tabela 2 - Parâmetros de simulação do Experimento 1

Simuladores	Mininet-Wifi e SUMO
Network Area	20Km²
Raio de cobertura RSU	800m(com sobreposição)
Propagation Model	Definido pelo trânsito do SUMO
Velocidade dos Veículos	**20,30,40,60,100 Km/h (todos os carros)**
Datagramas UDP	1470bytes
Número de estações	20
Número de RSUs	6
Antenna Model	OmniAntenna

Com base nesses parâmetros fizemos os experimentos. Assim, obtivemos os seguinte resultado para a vazão na Figura 18:

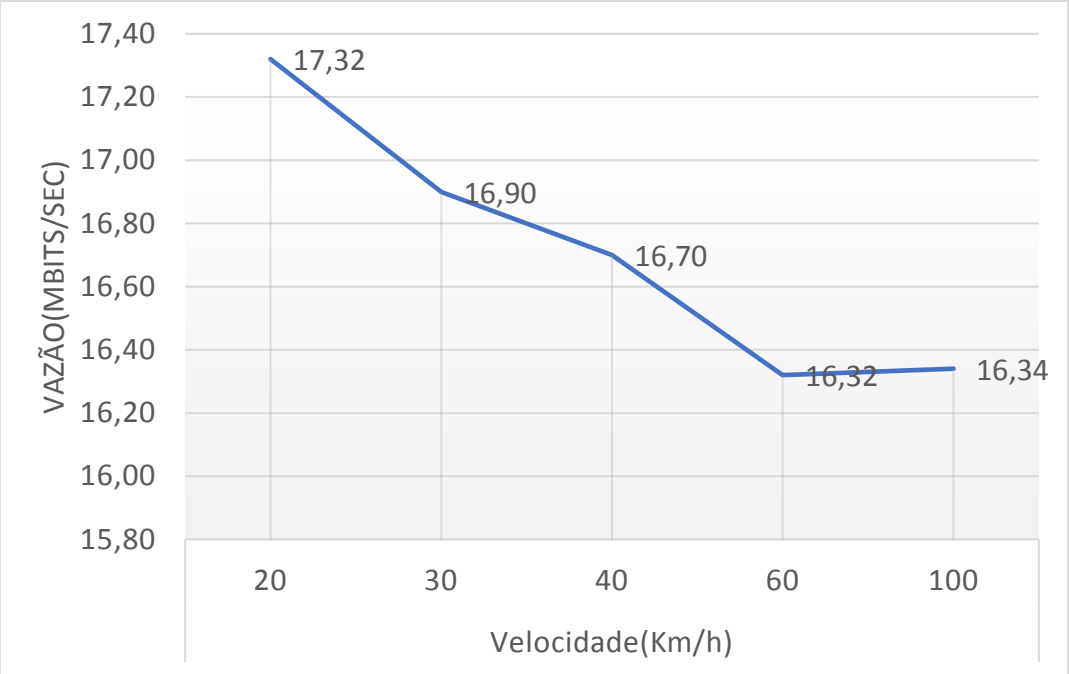

Figura 18 - Gráfico de Vazão - Experimento 1

Percebemos uma queda de 17,32Mbits/s para 16,32Mbits/s quando aumentamos a velocidade conforme o gráfico acima. Era o de se esperar que com o aumento da velocidade a vazão diminuísse.

O mesmo se dá, porém inversamente, com o jitter. Segue o gráfico na Figura 19:

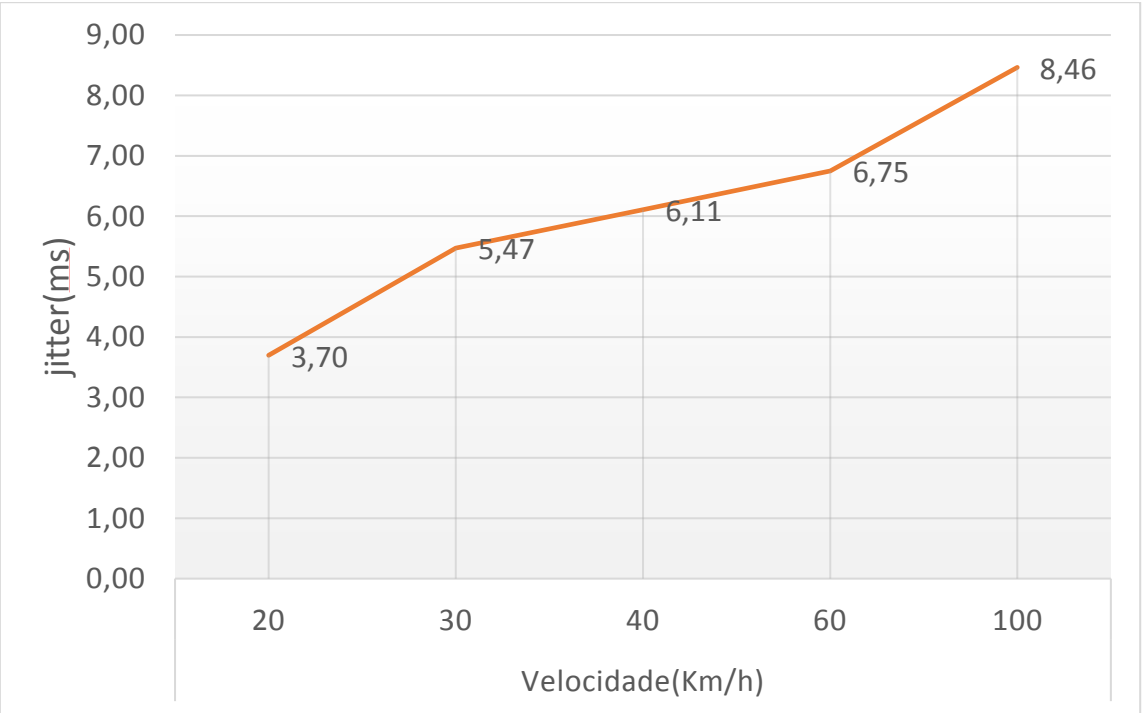

Figura 19 - Gráfico de Jitter - Experimento 1

Já o erro segue o mesmo padrão, se comparado ao Jitter. Vejamos na Figura 20:

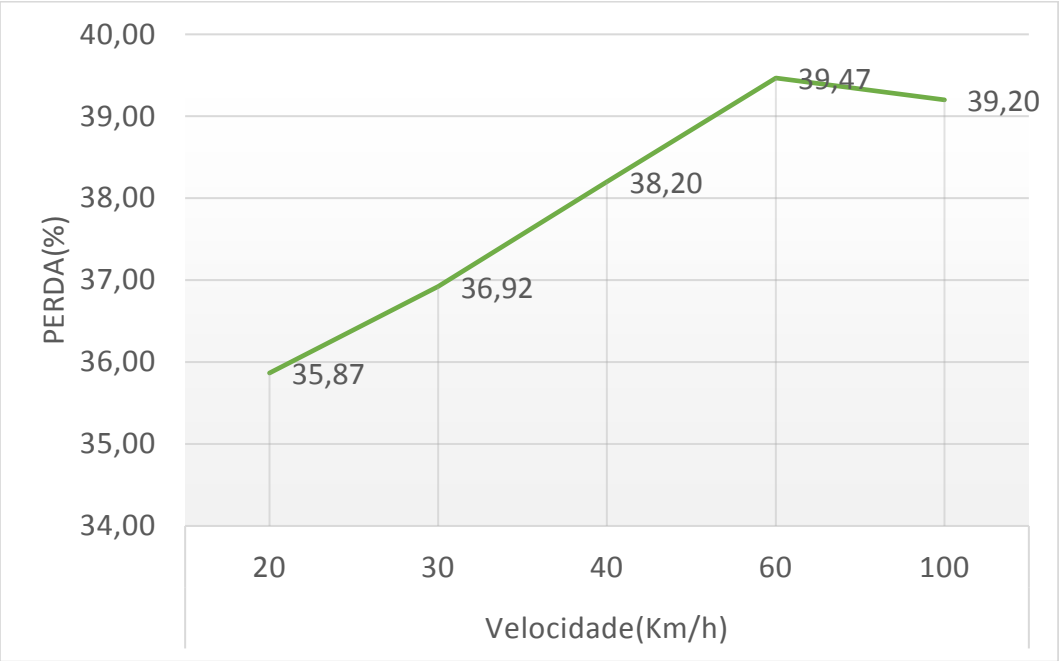

Figura 20 - Gráfico de Perda - Experimento 1

Estes gráficos mostraram os experimentos realizados onde foram feitas variações de velocidade dos nós(carros) da rede VANETs emulada. Na próxima seção 6.3 faremos a variação do número de carros.

6.3 Experimento 2: Análise do impacto da variação do número de carros

A seguir mostraremos os parâmetros de simulação utilizados nesta parte dos experimentos na Tabela 3:

Tabela 3 - Parâmetros de simulação do Experimento 2

Simuladores	Mininet-Wifi e SUMO
Network Area	20Km²
Raio de Cobertura da RSU	800m(com sobreposição)
Propagation Model	Definido pelo trânsito do SUMO
Velocidade dos Veículos	20Km/h
Datagramas	1470bytes
Número de estações(Veículos)	**10,20,30,40**
Número de RSUs	6
Antenna Model	OmniAntenna

Com base nesses parâmetros fizemos os experimentos. Assim, obtivemos os seguinte resultados para a vazão na Figura 21:

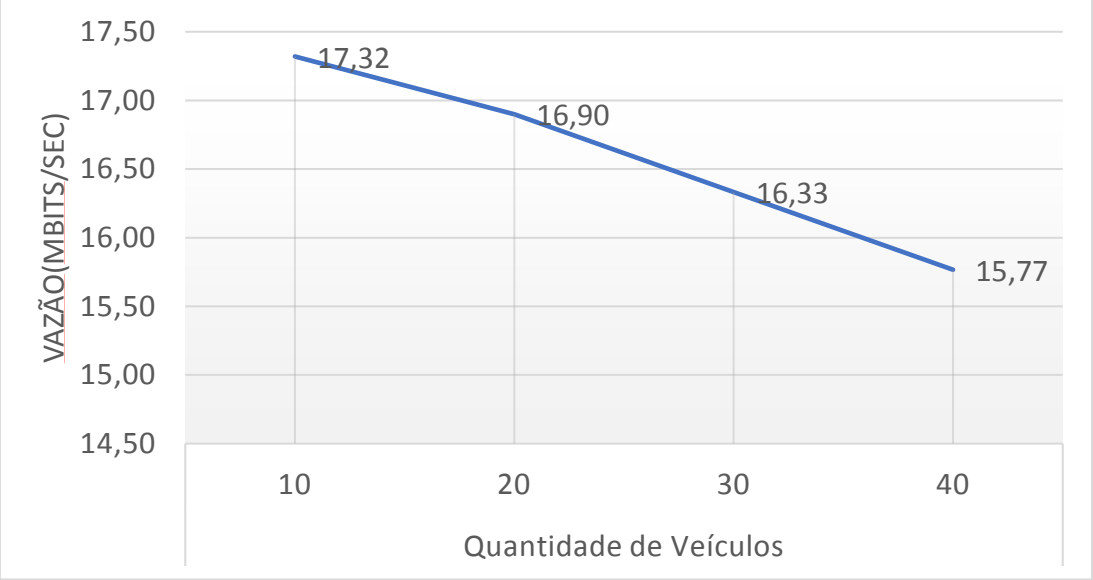

Figura 21 - Gráfico de Vazão - Experimento 2

Percebemos uma queda quase linear da vazão de 17,32Mbits/s para 15,77Mbits/s quando aumentamos a quantidade de nós conforme o gráfico acima. Era o de se esperar que com o aumento da quantidade de nós diminuísse a vazão.

O mesmo se dá com o jitter. Segue o gráfico na Figura 22:

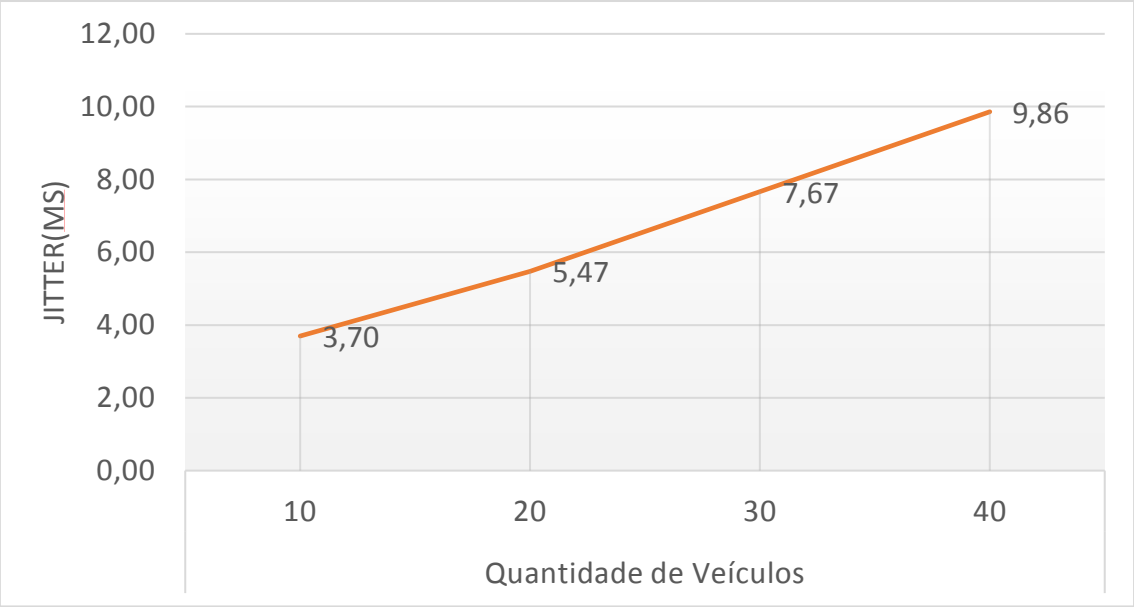

Figura 22 - Gráfico de Jitter - Experimento 2

Já o erro segue o mesmo padrão, se comparado ao Jitter. Vejamos na Figura 23:

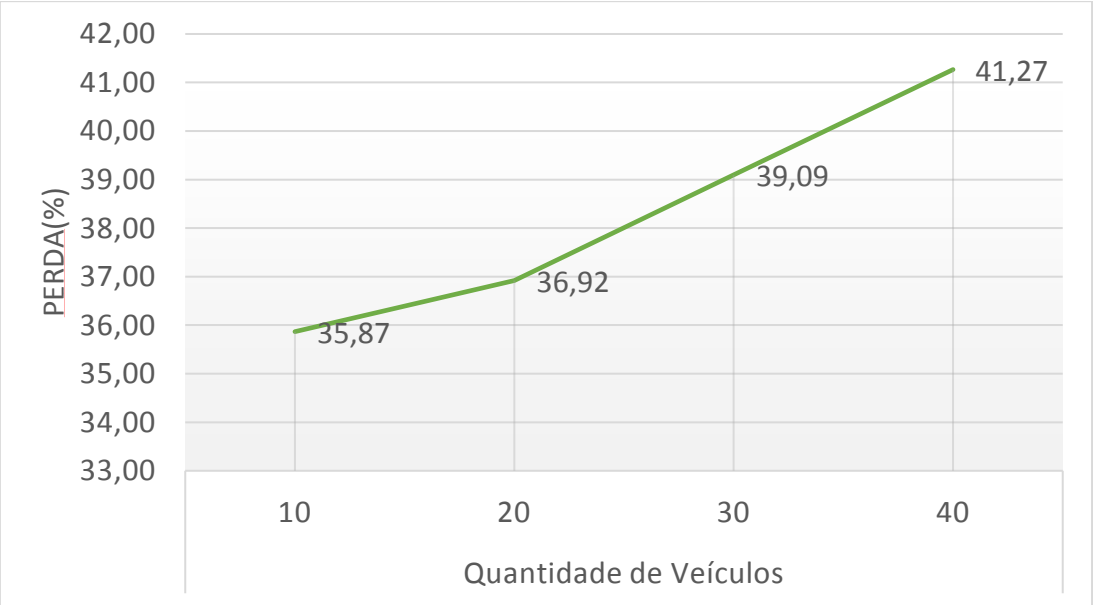

Figura 23 - Gráfico de Perda - Experimento 2

Estes gráficos mostraram os experimentos realizados onde foram feitas variações de velocidade dos nós(carros) da rede VANETs emulada. Na próxima seção 6.4 faremos a variação do raio de cobertura das RSUs.

6.4 Experimento 3: Análise do impacto da variação do raio de cobertura das RSUs

A seguir mostraremos os parâmetros fixos para simulação:

Tabela 4 - Parâmetros de simulação do Experimento 3

Simuladores	Mininet-Wifi e SUMO
Network Area	20Km²
Raio de Cobertura RSU	**300, 500, 800,1000,5000m**
Propagation Model	Definido pelo trânsito do SUMO
Velocidade dos Veículos	20 Km/h
Datagramas	1470bytes
Número de estações	20
Número de RSUs	6
Antenna Model	OmniAntenna

Com base nesses parâmetros fizemos os experimentos. Assim, obtivemos os seguinte resultados para a vazão na Figura 24:

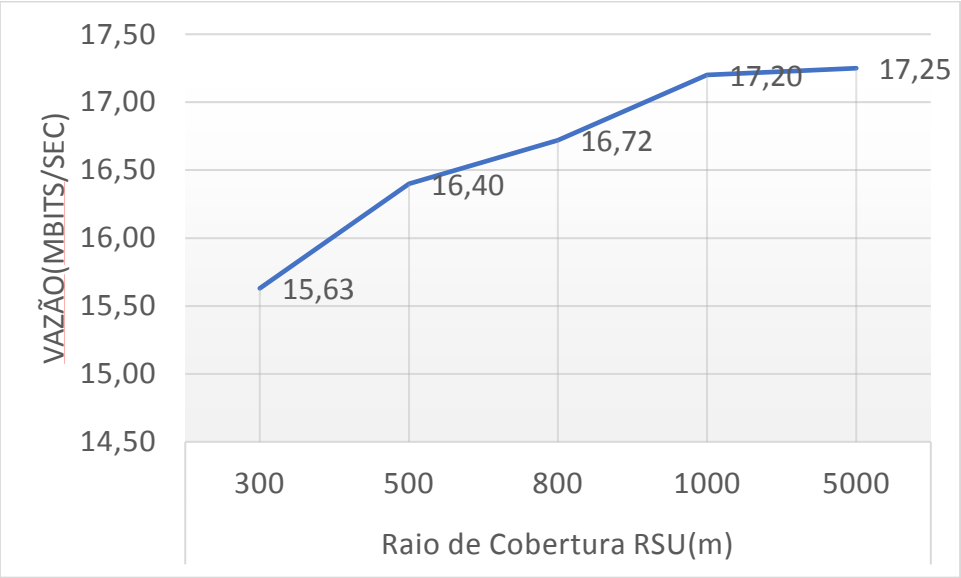

Figura 24 - Gráfico de Vazão - Experimento 3

Percebemos um aumento de 15,63Mbits/s para 17,25Mbits/s quando aumentamos o raio de cobertura conforme o gráfico acima. Era o de se esperar que com o aumento do raio de cobertura a vazão aumentasse

O mesmo se dá, porém inversamente, com o jitter. Segue o gráfico na Figura 25:

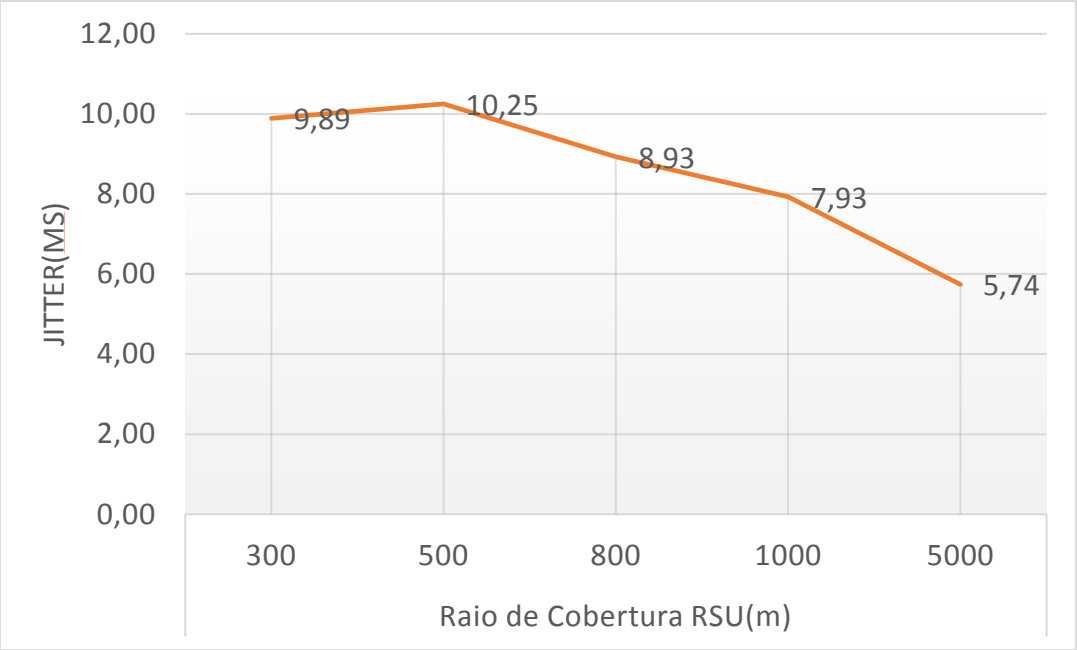

Figura 25 - Gráfico de Jitter - Experimento 3

Já o erro segue o mesmo padrão, se comparado ao Jitter. Vejamos na Figura 26:

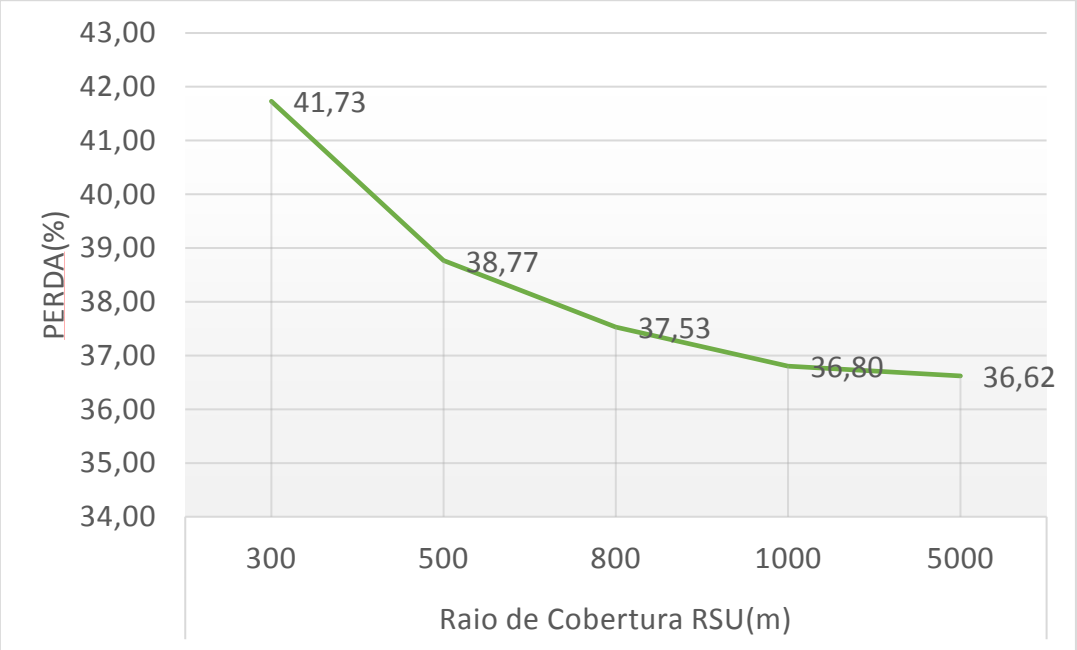

Figura 26 - Gráfico de Perda - Experimento 3

Estes gráficos mostraram os experimentos realizados onde foram feitas variações de velocidade dos nós(carros) da rede VANETs emulada. Na próxima seção, a 7, faremos as conclusões e menções de trabalhos futuros.

7 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi proposta uma análise de desempenho de rede com tráfegos reais em VANETs utilizando SDN. Essa análise foi composto por um mapa georreferenciado extraído do OpenStreetMap para servir como plano de coordenadas das carros(estações) e Pontos de Acesso(RSUs). Importamos este mapa no SUMO para gerar os tráfegos e aplicar as políticas e regras de trânsito. De posse do tráfego de veículos, sincronizamos os nós(estações) criados, via programação, no Mininet-Wifi para atualizamos as coordenadas dos nós. Os nós RSUs têm posições fixas no mapa.

De posse da integração entre SUMO e Mininet-Wifi, onde tivemos que sincronizar todos os nós do Mininet-Wifi com o SUMO, lendo, em tempo de execução, suas coordenadas, variamos os parâmetros velocidades dos veículos, range das RSUs e números de nós para avaliarmos a vazão, o jitter e as perdas de pacotes ocasionadas durante uma simulação de tráfego de veículos em zona urbana. Com os resultados, via gráficos, pudemos avaliar a viabilidade das redes VANETs num cenário urbano.

Os trabalhos apresentados no capítulo de trabalhos relacionados serviram para fomentar o desenvolvimento das redes VANETs. Estes trabalhos mostraram os desafios da implementação dos protocolos de comunicação, ofereceram propostas de novos modelos de gestão de dados e roteamento e expuseram os gargalos relacionados à dinamicidade da topologia desta rede emergente.

7.1 Contribuições

Neste trabalho, desenvolvemos uma emulação e não uma simulação de rede. A emulação fornece resultados mais próximos do ambiente real. Com isto, nossos resultados servirão de base para outros pesquisadores neste contexto de VANET definida por software.

7.2 Trabalhos Futuros

Como trabalhos futuros, estamos planejando obter um sistema VANET baseado em SDN hierárquico, com configurações de vários controladores SDN de maneira hierárquica. Os controladores de camada inferiores abrangerão uma área menor e estariam incumbidos por controlar o comportamento da rede para alterações dinâmicas e locais na topologia, enquanto menores mudanças

dinâmicas e globais seriam tratadas pelos controladores de camada mais alta, como trata (KU *et al.*, 2014).

8 BIBLIOGRAFIA

CAMPOS, M. B.; MARTINS, J. S. B. Uma Proposta de Arquitetura de Segurança para a Detecção e Reação a Ameaças em Redes SDN. *Revista Brasileira de Computação Aplicada - RBCA*, v. 9, n. 01, p. 1, 2017.

COSTA, J. *et al.* ˜ o de Dados em VANETs baseado Protocolo para Disseminac ¸ a em M ´ etricas de Redes Complexas : Um Estudo de Caso com Sistema de Gerenciamento de Trˆ ansito. 2016.

CUNHA, F. *et al.* Data communication in VANETs : Survey , applications and challenges. *Ad Hoc Networks*, v. 44, p. 90–103, 2016. Disponível em: <http://dx.doi.org/10.1016/j.adhoc.2016.02.017>.

DOS REIS FONTES, R. *et al.* From theory to experimental evaluation: Resource management in software-defined vehicular networks. *IEEE Access*, v. 5, p. 1–8, 2017.

FERNANDES, E. L.; ROTHENBERG, C. E. OpenFlow 1.3 Software Switch. *Anais do 32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos – SBRC*, p. 1021–1028, 2014.

KU, I. *et al.* Towards software-defined VANET: Architecture and services. *2014 13th Annual Mediterranean Ad Hoc Networking Workshop, MED-HOC-NET 2014*, p. 103–110, 2014.

LIN, D. *et al.* MoZo: A Moving Zone Based Routing Protocol Using Pure V2V Communication in VANETs. *IEEE Transactions on Mobile Computing*, v. 16, n. 5, p. 1357–1370, 2017.

SILVA, F. A. *et al.* Vehicular Networks: A New Challenge for Content-Delivery-Based

Applications. *ACM Computing Surveys*, v. 49, n. 1, p. 1–29, 2016. Disponível em: <http://dl.acm.org/citation.cfm?doid=2911992.2903745>.

SOUSA, A. M. DE; ARAÚJO, F. R. C.; SAMPAIO, L. N. Encaminhamento Seletivo de Interesses em Redes Veiculares de Dados Nomeados Baseado no Tempo de Vida do Enlace. *XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, 2018. Disponível em: <http://portaldeconteudo.sbc.org.br/index.php/sbrc/article/view/2461/2425>.

TREVIZAN DE OLIVEIRA, B.; BATISTA GABRIEL, L.; BORGES MARGI, C. TinySDN: Enabling multiple controllers for software-defined wireless sensor networks. *IEEE Latin America Transactions*, v. 13, n. 11, p. 3690–3696, 2015.

APÊNDICE 1 – CÓDIGO PYTHON PARA CRIAÇÃO DA REDE

```
#!/usr/bin/python
import os
from mininet.node import Controller,OVSKernelSwitch
from mininet.log import setLogLevel, info
from mn_wifi.node import OVSKernelAP,UserAP
from mn_wifi.cli import CLI_wifi
from mn_wifi.net import Mininet_wifi
from mn_wifi.sumo.runner import sumo
from mn_wifi.link import wmediumd, mesh
from mn_wifi.wmediumdConnector import interference
from time import sleep
from mininet.net import Mininet
carNumber = 10

def topology():
    "Create a network."
    net = Mininet_wifi(controller=Controller, accessPoint=OVSKernelAP, link=wmediumd, wmediumd_mode=interference)
    #net = Mininet_wifi(controller=Controller, accessPoint=UserAP, link=wmediumd, wmediumd_mode=interference)

    info("*** Creating nodes\n")
    cars = []
    for id in range(0, carNumber):
        cars.append(net.addCar('car%s' % (id+1), wlans=2))

    e1 = net.addAccessPoint('e1', ssid='vanet-ssid', mac='00:00:00:11:00:01', mode='g', channel='1',  position='3279.02,3736.27,0', range='55000')
    e2 = net.addAccessPoint('e2', ssid='vanet-ssid', mac='00:00:00:11:00:02', mode='g', channel='6',  position='2320.82,3565.75,0', range='55000')
    e3 = net.addAccessPoint('e3', ssid='vanet-ssid', mac='00:00:00:11:00:03', mode='g', channel='11', position='2806.42,3395.22,0', range='55000')
    e4 = net.addAccessPoint('e4', ssid='vanet-ssid', mac='00:00:00:11:00:04', mode='g', channel='1',  position='3332.62,3253.92,0', range='55000')
    e5 = net.addAccessPoint('e5', ssid='vanet-ssid', mac='00:00:00:11:00:05', mode='g', channel='6',  position='2887.62,2935.61,0', range='55000')
    e6 = net.addAccessPoint('e6', ssid='vanet-ssid', mac='00:00:00:11:00:06', mode='g', channel='11', position='2351.68,3083.40,0', range='55000')

    switch = net.addSwitch('switch',dpid='52000000000')
    c1 = net.addController('c1')
    server = net.addHost('server', ip = '192.168.0.250/24')

    info("*** Setting bgscan\n")
    #net.setBgscan(signal=-45, s_inverval=5, l_interval=10)
    net.setBgscan(signal=-90, s_inverval=5, l_interval=10)#new

    info("*** Configuring Propagation Model\n")
    #net.setPropagationModel(model="logDistance", exp=2)
    net.setPropagationModel(model="logDistance", exp=2)#new
    info("*** Configuring wifi nodes\n")
    net.configureWifiNodes()
    net.addLink(e1, switch)
    net.addLink(e2, switch)
    net.addLink(e3, switch)
    net.addLink(e4, switch)
    net.addLink(e5,switch)
    net.addLink(e6,switch)
    net.addLink(server,switch)
```

```
    info("*** Configuring wifi nodes\n")
    net.configureWifiNodes()
    net.addLink(e1, switch)
    net.addLink(e2, switch)
    net.addLink(e3, switch)
    net.addLink(e4, switch)
    net.addLink(e5,switch)
    net.addLink(e6,switch)
    net.addLink(server,switch)

    for car in cars:
        net.addLink(car, intf=car.params['wlan'][1],cls=mesh, ssid='mesh-ssid', channel=5)
    net.useExternalProgram(program=sumo, port=8813, config_file='map.sumocfg')
    info("*** Starting network\n")
    net.build()
    c1.start()
    switch.start([c1])
    e1.start([c1])
    e2.start([c1])
    e3.start([c1])
    e4.start([c1])
    e5.start([c1])
    e6.start([c1])

    for car in cars:
        car.setIP('192.168.0.%s/24' % (int(cars.index(car))+1),intf='%s-wlan0' % car)
        car.setIP('192.168.1.%s/24' % (int(cars.index(car))+1),intf='%s-mp1' % car)

    print("Play SUMO.")

    print("Execute o Servidor Iperf")
    CLI_wifi(net)
    print("Todos os carros, menos o car1, disparando pro servidor...")
    for x in range(carNumber):
        if (x != 0):
            print(x)
            print(carNumber)
            print(net.get('car%s'% (x+1)) )
            net.get('car%s'% (x+1)).cmd('iperf -c 192.168.0.250 -u -b 27M -y c -t 2000 &')
    for x in range(15):
        net.get('car1').cmd('iperf -c 192.168.0.250 -u -b 27M -y c -t 15 ')
        sleep(3)
    print("finalizando experimento...")
    CLI_wifi(net)
    info("Matando processos em segundo plano...")
    print("\n")
    for x in range(0,carNumber):
        if (x != 0):
            net.get('car%s'% (x+1)).cmd('kill %while')
            pid = int(net.get('car%s'% (x+1)).cmd('echo $!'))
            print(x+1)
            print(pid)
            net.get('car%s'% (x+1)).cmd('kill -9 %i' % pid)
    info("Todos os processos terminaram")
    info("*** Stopping network\n")
    net.stop()
```

APÊNDICE 2 – CÓDIGO DO CONTROLADOR RYU

```
from operator import attrgetter

from ryu.app import simple_switch_13
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER, DEAD_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.lib import hub


class SimpleMonitor13(simple_switch_13.SimpleSwitch13):

    def __init__(self, *args, **kwargs):
        super(SimpleMonitor13, self).__init__(*args, **kwargs)
        self.datapaths = {}
        self.monitor_thread = hub.spawn(self._monitor)

    @set_ev_cls(ofp_event.EventOFPStateChange,
                [MAIN_DISPATCHER, DEAD_DISPATCHER])
    def _state_change_handler(self, ev):
        datapath = ev.datapath
        if ev.state == MAIN_DISPATCHER:
            if datapath.id not in self.datapaths:
                self.logger.debug('register datapath: %016x', datapath.id)
                self.datapaths[datapath.id] = datapath
        elif ev.state == DEAD_DISPATCHER:
            if datapath.id in self.datapaths:
                self.logger.debug('unregister datapath: %016x', datapath.id)
                del self.datapaths[datapath.id]

    def _monitor(self):
        while True:
            for dp in self.datapaths.values():
                self._request_stats(dp)
            hub.sleep(10)

    def _request_stats(self, datapath):
        self.logger.debug('send stats request: %016x', datapath.id)
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser

        req = parser.OFPFlowStatsRequest(datapath)
        datapath.send_msg(req)

        req = parser.OFPPortStatsRequest(datapath, 0, ofproto.OFPP_ANY)
        datapath.send_msg(req)

    @set_ev_cls(ofp_event.EventOFPFlowStatsReply, MAIN_DISPATCHER)
    def _flow_stats_reply_handler(self, ev):
        body = ev.msg.body

        self.logger.info('datapath         '
                         'in-port  eth-dst           '
                         'out-port packets  bytes')
        self.logger.info('---------------- '
                         '-------- ----------------- '
                         '-------- -------- --------')
        for stat in sorted([flow for flow in body if flow.priority == 1],
                           key=lambda flow: (flow.match['in_port'],
                                             flow.match['eth_dst'])):
            self.logger.info('%016x %8x %17s %8x %8d %8d',
                             ev.msg.datapath.id,
                             stat.match['in_port'], stat.match['eth_dst'],
                             stat.instructions[0].actions[0].port,
                             stat.packet_count, stat.byte_count)

    @set_ev_cls(ofp_event.EventOFPPortStatsReply, MAIN_DISPATCHER)
    def _port_stats_reply_handler(self, ev):
        body = ev.msg.body

        self.logger.info('datapath         port     '
                         'rx-pkts  rx-bytes rx-error '
                         'tx-pkts  tx-bytes tx-error')
        self.logger.info('---------------- -------- '
                         '-------- -------- -------- '
                         '-------- -------- --------')
        for stat in sorted(body, key=attrgetter('port_no')):
            self.logger.info('%016x %8x %8d %8d %8d %8d %8d %8d',
                             ev.msg.datapath.id, stat.port_no,
                             stat.rx_packets, stat.rx_bytes, stat.rx_errors,
                             stat.tx_packets, stat.tx_bytes, stat.tx_errors)
```